PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Représentation abstraite pour la génération automatique de scènes par planification spatiale
Auteur: Author:	Guillaume Bouilly
Date:	2012
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Bouilly, G. (2012). Représentation abstraite pour la génération automatique de scènes par planification spatiale [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/843/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/843/
Directeurs de recherche: Advisors:	Michel Gagnon, & Benoît Ozell
Programme: Program:	Génie informatique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

REPRÉSENTATION ABSTRAITE POUR LA GÉNÉRATION AUTOMATIQUE DE SCÈNES PAR PLANIFICATION SPATIALE

GUILLAUME BOUILLY
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AVRIL 2012

© Guillaume Bouilly, 2012.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

REPRÉSENTATION ABSTRAITE POUR LA GÉNÉRATION AUTOMATIQUE DE SCÈNES PAR PLANIFICATION SPATIALE

présenté par : BOUILLY, Guillaume
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury d'examen constitué de :

M. PESANT, Gilles, Ph.D., président.
M. GAGNON, Michel, Ph.D., membre et directeur de recherche.
M. OZELL, Benoit, Ph.D., membre et codirecteur de recherche.
M. BOYER, François-Raymond, Ph.D., membre.

REMERCIEMENTS

J'aimerais premièrement remercier le CRSNG, le FQRNT, Ubisoft et CAE pour leur support financier qui m'a permis de compléter ma maîtrise. J'aimerais également remercier mes directeurs de recherche, Michel Gagnon et Benoit Ozell, pour leur encadrement et leur aide tout au long de mon cheminement. Ensuite, j'aimerais remercier mes collègues Paul Gédéon, Jonathan Tardif, François-Xavier Desmarais et Maxime Ouellet de m'avoir accompagné durant cette aventure. De plus, j'aimerais remercier mon superviseur chez Eidos Montréal, Philippe Leblanc, pour m'avoir accordé une journée de travail par semaine durant les derniers mois de mon projet. Finalement, j'aimerais remercier ma famille et mes proches pour tout leur support moral durant ces deux années de dur labeur, principalement durant les dernières semaines de rédaction.

RÉSUMÉ

La création d'environnements 3D constitue une tâche manuelle ardue qui demande beaucoup de temps et qui gagnerait à être automatisée. La *planification spatiale* regroupe des techniques reposant sur la programmation par contraintes qui visent à placer correctement des objets à l'intérieur d'une scène statique. Les principales lacunes de ces techniques relèvent du manque de flexibilité introduit par les formats d'entrée et les représentations sémantiques utilisées.

Le but de notre projet de recherche est d'évaluer la possibilité de définir un format de représentation abstraite pouvant être utilisé dans le cadre du placement spatial. Notre recherche repose sur la supposition qu'il est possible de définir une abstraction des concepts intervenant dans le placement spatial, et que cette représentation pourrait représenter un problème résoluble.

Nous proposons une solution basée sur un format de représentation abstraite et une ontologie. La représentation abstraite permet de spécifier des configurations dans lesquelles des objets sont impliqués dans des actions selon des modalités spatiales définies. L'ontologie définit une hiérarchie des objets, contrôleurs et actions impliqués dans le placement spatial, ainsi que des relations permettant d'étendre ou de caractériser la représentation abstraite.

Notre expérimentation repose sur la génération aléatoire d'un grand nombre de descriptions de scènes abstraites. Ces descriptions sont générées de manière à contenir un nombre croissant d'objets et de relations, pour identifier les limites de notre solution. Au terme de notre expérimentation, nous avons généré des scènes impliquant jusqu'à 15 objets et 16 relations.

L'analyse de nos résultats nous a permis de confirmer qu'il est possible d'utiliser une représentation abstraite pour la génération automatique de scènes par placement spatial. Les limitations de notre modélisation nous ont permis d'orienter les travaux futurs vers l'optimisation de la résolution à partir de notre modélisation, la modélisation d'une ontologie de grande envergure pour le placement spatial et l'évaluation de l'utilisation d'une représentation abstraite dans des problèmes plus complexes. La réalisation de ces travaux faciliterait la création de scènes 3D dans divers domaines d'applications multimédia tels que le cinéma ou les jeux vidéo.

ABSTRACT

The creation of 3D environments is a tedious and costly task. *Space Layout Planning* techniques aim to use constraint programming in order to automatically assign a spatial position to the objects of a 3D scene. We aim to address the lack of flexibility of these techniques in regards to the input formats and the semantic representations.

The goal of our research project is to create a new space layout planning input format that would offer a higher level of abstraction. This goal is based on the supposition that it is possible to express the concepts of a space layout planning problem as an abstraction, and that this abstraction can be transformed into a solvable constraint satisfaction problem.

Our solution involves using an abstract representation input format and an ontology. The abstract representation can be used to define multiple configurations, which describe several objects that can take part into actions based on clearly defined spatial modalities. The ontology's role is to define a hierarchy of objects that can be used in space layout planning, as well as several relations that can complement the abstract representation.

Our experimentation relies on a random generation of many abstract representations. Each step of the generation algorithm involves a growing number of objects and a growing number of relations. The goal of this experimental approach is to identify the limits of our solution in regards to the query's complexity. At the end of our experimentation, we were able to generate scenes with a maximal number of 15 objects and 16 relations.

We have been able to confirm that space layout planning based on an abstract input format is a possible approach for the resolution of space layout planning problems in order to automatically generate 3D scenes. Future work on this approach involves the optimisation of the problem's modeling, the creation of a larger ontology for space layout planning, and the evaluation of an abstract representation with more complicated space layout problems. The completion of this future work would address the limitations of our current approach, and could be the key to the application of space layout planning techniques for the automatic creation of 3D scenes for movies or video games.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

GITAN	Grammaire pour l'Interprétation de Texte et d'ANimations
CSP	Constraint Satisfaction Problem
AABB	Axis Aligned Bounding Box
OBB	Oriented Bounding Box
RDF	Resource Description Framework
OWL	Web Ontology Language
SPARQL	SPARQL Protocol and RDF Query Language
GUM	Generalized Upper Model

CHAPITRE 1

INTRODUCTION

La fin du XX^{e} siècle a vu l'apparition de techniques multimédia permettant la création d'environnements 3D. Quelques décennies plus tard, ces environnements sont grandement répandus dans la vie de tous les jours, que ce soit dans les jeux vidéo, le cinéma, les sites internet ou les diverses applications que l'on peut télécharger sur notre téléphone.

La création de tels environnements constitue une tâche de grande envergure. Pour cette raison, certains projets de recherche se sont intéressés à l'automatisation de certaines étapes de la création d'une scène 3D. Parmi les multiples techniques permettant la génération automatique d'une scène virtuelle, nous nous intéressons à la planification spatiale. Le but de cette technique est d'automatiser le positionnement des objets dans un environnement 3D.

Il a été démontré que la planification spatiale constitue une solution réalisable, et il existe plusieurs projets de recherche appliquant cette technique à la génération automatique d'une scène virtuelle. Par contre, nous pensons qu'il est encore possible d'apporter des améliorations à cette approche.

Le but de notre projet de recherche est de contribuer à la résolution du problème de placement spatial dans une scène multimédia. Les aspects principaux que nous voulons améliorer sont la représentation initiale des objets à placer et la représentation de la connaissance venant assister les méthodes de résolution.

Nous proposons un formalisme de représentation de la scène à réaliser qui se situe à un niveau d'abstraction supérieur aux solutions déjà existantes. De plus, nous proposons un format de représentation des connaissances pour la planification spatiale qui nous permet de compléter cette représentation abstraite.

Ce projet de recherche est réalisé dans le cadre du projet GITAN (Grammaire pour l'Interprétation de Texte et d'ANimations). Le but de GITAN est de parvenir à traduire automatiquement du texte en animations 3D. Sa structure suit un pipeline constitué de différentes étapes de conversions dans le but de passer du texte à l'animation. Les trois grandes étapes de ce pipeline sont l'analyse de texte, la traduction du domaine de la langue au domaine graphique, et la construction de l'animation 3D.

Dans le cadre de GITAN, notre projet de recherche constitue la première étape de la construction de l'animation 3D. Nous supposons que le traitement de la langue a été effectué avec succès, et nous ne visons pas une qualité visuelle optimale en ce qui a trait à l'infographie. Nous visons plutôt à utiliser des volumes englobants pour le positionnement

des objets. Le rôle du projet est de produire une scène statique contenant l'ensemble des éléments de l'animation initialisés à leur position initiale. Cette scène doit être accompagnée de l'information sémantique nécessaire à la production de l'animation.

Le chapitre 2 présente tout d'abord la problématique associée au placement d'objets dans des scènes multimédia. Par la suite, nous énonçons les hypothèses et les objectifs de notre projet de recherche.

Le chapitre 3 constitue la revue de littérature. Nous y présentons l'état de l'art en ce qui a trait à la planification spatiale, plus précisément par rapport aux méthodes de résolutions, aux formats d'entrées et à l'utilisation de représentation de la connaissance pour la planification spatiale.

Le chapitre 4 présente la solution que nous proposons pour répondre à notre question de recherche. Celle-ci couvre le formalisme de représentation abstraite, le format de représentation de la connaissance, ainsi que la modélisation du problème de contraintes. La méthodologie qui nous a mené à cette solution est couverte dans le chapitre 5.

Le chapitre 6 présente les résultats obtenus à l'aide de notre solution ainsi que la discussion associée à ces résultats. Finalement, le chapitre 7 présente les orientations de recherches que nous proposons suite à l'analyse de nos résultats.

CHAPITRE 2

PROBLÉMATIQUE

Ce chapitre présente la problématique associée à la création automatique de scènes virtuelles par planification spatiale. Nous exposons celle-ci en deux parties, en situant d'abord l'état actuel de la création de scène à l'aide des outils traditionnels, puis en présentant certaines pistes d'automatisation visant à résoudre le problème. De cette problématique découlent la question de recherche et le but de notre projet, suivis de nos hypothèses et de nos objectifs.

2.1 Situation actuelle

Depuis leur première apparition dans l'univers du multimédia, la création d'une animation 3D a impliqué la réalisation de plusieurs étapes de production artistique. Dans un premier temps, il est nécessaire de modéliser les géométries 3D qui vont constituer les éléments de l'animation. Dans le cas où les objets sont dynamiques, leur animation implique de prédéfinir les mouvements que ces objets seront capables de réaliser. Finalement, pour qu'une scène virtuelle constitue un environnement réaliste, il est important que tous les objets qu'elle présente soient disposés correctement. Une disposition correcte implique de respecter les règles et habitudes du monde que l'on souhaite représenter.

Les dernières années ont été marquées par des avancées remarquables quant au niveau de réalisme pouvant être atteint par les animations 3D. Nous sommes maintenant capables de produire des animations 3D en haute définition, où la qualité visuelle et la fluidité des animations se rapprochent grandement de la réalité. Étant donné de tels résultats, nous pourrions penser que les animations 3D ont atteint leur sommet, et qu'il n'est plus nécessaire de viser à les améliorer.

Par contre, l'atteinte d'un tel niveau de réalisme a un effet négatif sur la création d'animations. En effet, des environnements virtuels de haute qualité impliquent un niveau de précision beaucoup plus élevé. Il n'est donc pas surprenant que les étapes de modélisation, d'animation et de positionnement se complexifient, et demandent une charge de travail beaucoup plus importante qu'il y a quelques années. Tutenel *et al.* (2008) compare l'état actuel de la création d'environnement 3D à une forme d'artisanat de haute technologie.

Si l'on s'intéresse seulement à la disposition des objets dans une scène, on se rend compte rapidement de la complexité de la tâche. En effet, le placement d'objets dans une scène vir-

tuelle implique de manipuler un objet dans un environnement 3D, ce qui implique six degrés de liberté. Or, pour manipuler les objets dans cet environnement, nous sommes limités à des outils qui n'offrent que deux degrés de liberté. Les limitations des méthodes de positionnement, combinées aux besoins de précisions des récentes améliorations ne font que grandir le besoin de solutions pour faciliter la création d'environnement virtuels.

2.2 Pistes de solution

La technique la plus répandue pour réduire les limitations associées au positionnement d'objets est d'offrir un environnement de manipulation basé sur 3 angles de vue orthogonaux, simulant ainsi six degrés de liberté. Une autre approche a été de développer de nouveaux outils de manipulation offrant jusqu'à six degrés de liberté. Les limitations de ces approches sont expliquées en détail par Xu (2001).

Un moyen plus attrayant de faciliter la création de contenu multimédia a fait l'objet de quelques projets de recherche dans les dernières années. Il s'agit d'automatiser les étapes du processus de création. Ces techniques peuvent être appliquées à de nombreuses étapes de la création de scènes virtuelles, comme la génération de reliefs, de modèles de végétation ou de milieux urbains, ainsi que l'assignation automatique d'une position aux objets d'une scène.

La définition d'un problème de planification spatiale implique de spécifier des objets à placer, ainsi que des relations à respecter pour le placement de ces objets. La résolution d'un problème de planification spatiale implique d'attribuer une position correcte et réaliste à chacun des objets à placer.

Évidemment, l'automatisation d'un processus complexe constitue en soi un problème complexe. Étant donné le grand nombre d'objets pouvant être présents dans une scène 3D, la planification spatiale implique un très grand nombre de variables. De plus, l'ensemble des configurations possibles avec l'ensemble des objets considérés grandit exponentiellement.

Au fil des années, des algorithmes reposant sur la programmation par contraintes ont été développés pour obtenir des solutions correctes au problème de planification spatiale. Il s'agit donc d'une technique réalisable et applicable à la création automatique de scènes virtuelles. Cependant, il reste encore des difficultés non résolues dues à la difficulté du problème.

La principale lacune des solutions de planification spatiale relève de la représentation initiale du problème. En effet, le problème implique rapidement un très grand nombre d'objets et de relations, et il est difficile de représenter tous ces éléments dans un format facilement compréhensible. À ce jour, il n'existe pas de représentation reconnue pour la représentation initiale de la planification spatiale. Cela est dû au fait que les efforts ont plutôt été dirigés vers la résolution du problème.

L'une des principales difficultés de la planification spatiale relève de la nécessité de recourir à une forme de connaissance pour effectuer le placement des objets. De manière générale, un être humain s'attend à ce que le positionnement des objets dans un environnement virtuel respecte les conventions de la réalité. Bien que la plupart des approches de planification spatiale utilisent une forme de connaissance, celles-ci sont loin d'être parfaitement adaptées à la planification spatiale dans des contextes variés.

Bien qu'il soit possible de résoudre un problème de planification spatiale, la plupart des projets ne considèrent pas la possibilité de tourner les objets sur eux-mêmes, et ne couvrent donc pas les six degrés de liberté à combler. Une solution complète permettrait non seulement d'attribuer une position correcte aux objets, mais également de les orienter correctement.

Finalement, il n'existe aucune métrique d'évaluation relative aux scènes produites par planification spatiale. Bien qu'il existe plusieurs projets de recherche ayant réussi à générer des scènes par le biais de la planification spatiale, aucun d'entre eux ne propose de métriques d'évaluation. Pour cette raison, il est difficile de comparer les approches entre elles.

2.3 But

Dans le cadre GITAN, le but de notre projet est d'être capable de générer une scène statique disposée correctement pour y générer les animations décrites dans le texte à animer. Sachant que la planification spatiale est une solution valide à ce problème, nous visons donc à appliquer et améliorer cette technique pour la production de nos scènes.

La traduction à partir d'un texte exige que notre application de la planification spatiale supporte un haut niveau d'abstraction. En effet, le texte comporte généralement très peu d'informations spécifiques par rapport à la disposition des objets. Ces détails sont plutôt laissés à l'imagination ou aux connaissances de l'utilisateur.

Pour cette raison, nous avons choisi de nous concentrer sur la représentation de départ et le traitement de la connaissance. Notre but est d'exprimer le problème de la planification spatiale par le biais d'une représentation offrant un grand niveau d'abstraction, et de combler le manque d'information de cette abstraction par l'utilisation d'une représentation de la connaissance adaptée au problème de planification spatiale.

Nous aimerions également être capable d'évaluer les scènes que nous produisons, dans le but de pouvoir nous comparer aux techniques ne reposant pas sur une représentation abstraite. Pour cette raison, nous sommes intéressé à définir des métriques d'évaluation des scènes produites par planification spatiale.

L'expression de ces motivations nous mène à la formulation de la question de recherche suivante :

Est-il possible de générer automatiquement une scène virtuelle par planification spatiale à partir d'une représentation abstraite ?

2.4 Hypothèses

Notre hypothèse principale découle directement de notre question de recherche. Nous posons l'hypothèse qu'il est possible de créer automatiquement des scènes virtuelles par planification spatiale à partir d'une représentation abstraite. Cette automatisation consiste à représenter l'ensemble des informations contenues dans la représentation abstraite sous la forme d'une scène 3D visualisable.

Nous décomposons cette hypothèse selon les sous-hypothèses suivantes :

H1 Il est possible de créer une représentation abstraite contenant toutes les informations nécessaires à la planification spatiale ;

H2 Il est possible d'ajouter du contenu à la représentation abstraite à partir d'une représentation de connaissance ;

H3 Il est possible de transformer la représentation abstraite en une modélisation de problème de contraintes résoluble.

2.5 Objectifs

L'objectif global de notre projet de recherche est de répondre à notre question de recherche et à notre hypothèse principale. Notre objectif global est donc d'automatiser la création de scènes virtuelles par planification spatiale à partir d'une représentation abstraite.

Cet objectif peut être décomposé selon les objectifs spécifiques suivants :

O1 Identifier les informations essentielles au problème de planification spatiale ;

O2 Identifier un format de représentation abstraite de scène ;

O3 Identifier une représentation sémantique adaptée à la planification spatiale ;

O4 Vérifier la faisabilité de la planification spatiale à partir d'une représentation abstraite ;

O5 Définir des métriques d'évaluation de scènes produites par planification spatiale ;

O6 Déterminer la complexité maximale des scènes pouvant être traitées par cette approche.

CHAPITRE 3

REVUE DE LITTÉRATURE

Ce chapitre présente une revue de littérature de la recherche relative à la planification spatiale. Nous présentons tout d'abord une définition de la planification spatiale ainsi que ses principaux contextes d'applications. Par la suite, nous exposons la méthode de résolution ainsi que la théorie nécessaire pour sa compréhension. Nous terminons avec une revue des formats d'entrée et de représentation de la connaissance utilisés conjointement avec la planification spatiale.

3.1 Définition

La planification spatiale consiste à assigner une position numérique à un ensemble d'objets dans un espace donné, tout en respectant une série de contraintes pouvant affecter la manière de positionner un ou plusieurs objets. Le problème a fait son apparition dans les années soixante et a été plus récemment adapté au domaine de l'infographie. Homayouni (2000) présente une revue des recherches autour de ce problème de 1965 à 2000.

3.1.1 Placement architectural

Les premières références à la planification spatiale relèvent du placement architectural. Seehof *et al.* (1966) tente d'automatiser la tâche d'effectuer le placement de plusieurs unités de travail différentes sur le même étage d'une usine. Pfefferkorn (1975) s'intéresse au placement du mobilier dans un bureau. Il est le premier à formaliser la définition du problème.

Sa définition implique un ensemble de pièces vides, un ensemble d'objets à placer et un ensemble de contraintes. Les contraintes limitent le placement des objets à l'intérieur des pièces. Les contraintes indiquent par exemple que les objets doivent occuper des espaces disjoints, ou qu'il est nécessaire de laisser des espaces libres pour leur utilisation.

3.1.2 Applications à l'infographie

Plus récemment, des applications de ce problème au domaine de l'infographie ont été proposées. la planification spatiale dans l'espace est parfaitement adaptée pour représenter le placement d'objets 3D dans une scène virtuelle. Cependant, l'application au domaine de l'infographie nécessite une généralisation, car les scènes représentent des environnements en trois

dimensions. Précédemment, le problème avait été limité au placement en deux dimensions à partir d'un point de vue vertical.

3.2 Méthodes de résolution

Les premières tentatives de planification spatiale ont exploré des techniques basées sur des algorithmes aléatoires (Seehof *et al.* (1966)), ou sur des modèles heuristiques (Pfefferkorn (1975)). Assez rapidement, la programmation par contraintes s'est imposée comme étant la méthode de résolution la plus efficace pour ce problème. Baykan et Fox (1991) et Charman (1993) démontrent la supériorité de cette méthode de résolution.

Parallèlement à la programmation par contraintes, certains systèmes utilisent également une simulation physique pour renforcer l'aspect réaliste des scènes produites.

3.2.1 Programmation par contraintes

Cette section couvre des aspects théoriques associés à la programmation par contraintes. Nous présentons tout d'abord une définition de la programmation par contraintes, suivie d'un survol des étapes de modélisation et résolution du problème.

Définition

Apt (2003) définit un problème de satisfaction de contraintes (CSP) comme un ensemble de variables et un ensemble de contraintes sur ces variables. Chaque variable est caractérisée par un domaine de valeurs. Une contrainte constitue une restriction sur les combinaisons de valeurs prises simultanément par un ensemble de variables.

Une solution à un CSP constitue un sous-ensemble de valeurs légales pour chaque variable du problème, de manière à ce que toutes les contraintes soient satisfaites. Un CSP possédant au moins une solution est consistant. Inversement, un CSP ne possédant aucune solution est inconsistant.

Modélisation d'un CSP

La modélisation du problème consiste à définir l'ensemble des variables et leur domaine, ainsi que l'ensemble des contraintes sur ces variables. La modélisation du problème est une étape très importante de la programmation par contraintes et ne doit pas être négligée. En effet, il existe souvent plusieurs modèles pouvant représenter un même problème. Les choix de représentation des variables ou des contraintes peuvent avoir un fort impact sur l'efficacité ou le succès de la méthode de résolution.

Résolution d'un CSP

Il existe deux options pour la résolution d'un CSP : les méthodes spécifiques et les méthodes générales.

Les méthodes spécifiques constituent une implémentation spécialisée de certains algorithmes. Elles visent à optimiser la recherche de solution selon les spécificités du domaine d'application. Dans le domaine de la planification spatiale, les solutions reposent principalement sur des algorithmes spécifiques.

Les méthodes générales utilisent des méthodes d'inférence visant à réduire l'espace de recherche relié au problème. Ces méthodes reposent sur le concept de propagation des contraintes.

La résolution d'un CSP peut avoir trois objectifs :

- Déterminer si le problème étudié possède une solution
- Trouver une ou toutes les solutions du problème
- Trouver la ou les solutions optimales selon un critère

Dans le cadre de la planification spatiale, il est difficile d'évaluer toutes les solutions du problème. De manière générale, l'obtention d'une solution constitue une configuration acceptable de la scène. Il est également possible de produire plusieurs solutions et de les noter selon certains critères.

Actuellement, il est commun de viser à obtenir une seule solution valide. Pour cette raison, il n'existe pas de métriques d'évaluation des scènes produites par planification spatiale. Xu *et al.* (2002) sont les seuls à avoir chiffré une partie de leurs résultats. Leur système, CAPS, est capable de générer une scène comprenant 300 objets en 10 minutes.

Propagation des contraintes

Le concept le plus important relié à la résolution d'un CSP est la notion de propagation des contraintes. Bessiere (2006) définit la propagation des contraintes comme l'ensemble de tout raisonnement consistant à interdire des valeurs ou des combinaisons de valeurs pour certaines variables. Ces interdictions découlent de l'impossibilité de satisfaire certaines contraintes en fonction de certaines valeurs.

Plus concrètement, la propagation des contraintes consiste à réduire l'ensemble des valeurs possibles pour chaque variable à partir des contraintes du problème. Cela consiste aussi à réduire progressivement les valeurs possibles à partir des valeurs attribuées dans l'algorithme de recherche. La propagation des contraintes relève habituellement d'un algorithme avec retour en arrière, ce qui permet de corriger les erreurs lorsqu'un état sans solution possible est atteint.

Contraintes globales

van Hoeve et Katriel (2006) définissent une contrainte globale comme une contrainte définissant une relation s'appliquant à un nombre non-fixe de variables. L'exemple le plus connu de contrainte globale est la contrainte $alldifferent(x_1, ..., x_n)$. Cette contrainte spécifie qu'une même valeur ne peut être assignée à plus d'une variable soumise par la contrainte.

Une contrainte globale constitue généralement une forme de redondance au niveau de la formulation des contraintes. En effet, il serait possible d'utiliser un ensemble de contraintes binaires pour représenter la même relation. Dans le cas de la contrainte *alldifferent*, il aurait été possible d'exprimer la même relation à l'aide d'une contrainte de non-égalité appliquée sur chaque paire de variables.

L'utilisation d'une seule contrainte globale ne sert pas seulement à réduire le nombre de contraintes à exprimer lors de la modélisation du problème. En effet, lorsqu'une relation est définie à l'aide d'une contrainte globale, il est possible d'utiliser des algorithmes spécialisés pour la propagation des contraintes. Pour cette raison, Apt (2003) insiste sur le fait que la modélisation avec des contraintes globales est plus efficace que la modélisation avec plusieurs contraintes simples.

Le concept des contraintes globales est également applicable à la planification spatiale. La contrainte *geost*, définie par Beldiceanu *et al.* (2007), permet de s'assurer que l'ensemble des objets à placer ne s'interpénètrent pas.

Hiérarchisation de contraintes

Dans le cas de problèmes fortement contraints, il est possible qu'il n'existe aucune solution satisfaisant l'ensemble des contraintes. Dans de tels cas, il peut être plus souhaitable d'obtenir une solution qui ne respecte pas l'ensemble de contraintes, plutôt que de n'obtenir aucune solution.

La hiérarchisation de contraintes permet de définir des contraintes qui peuvent être violées pour la production d'une solution. La recherche d'une solution vise alors à trouver la solution qui satisfait le mieux possible la modélisation. Meseguer *et al.* (2006) présente les principales méthodes pour la création de contraintes qui peuvent être soulagées.

3.2.2 Simulation physique

Dans les systèmes incorporant une simulation physique, la solution obtenue par programmation par contraintes est soumise à l'effet de la gravité. De cette manière, la configuration finale des objets correspond à un équilibre physiquement possible, en plus de respecter les contraintes appliquées à la scène.

Shinya et Forgue (1995) introduisent avec leur système de placement un engin physique pour la détection de collisions et la dynamique de corps rigides. L'objectif est de détecter les collisions le plus rapidement possible, et d'utiliser des approximations pour la dynamique au niveau des objets.

Xu *et al.* (2002) réutilisent l'engin physique introduit par Shinya et Forgue (1995). Cet engin leur permet d'assurer la non interpénétration des objets, en plus d'évaluer une position stable pour chaque objet à partir d'un modèle de friction. Chaque objet est placé en chute libre au dessus de la position qui lui est attribuée. Sa position finale est déterminée une fois que son mouvement est arrêté. Le placement d'objets par chute libre est applicable dans le cadre des scènes considérées. En effet, le système s'intéresse au placement dans des scènes intérieures, et n'impliquent que des objets placés sur des surfaces horizontales.

3.3 Représentation des variables

Les variables de la planification spatiale sont toujours les objets à placer. Cependant, il est possible de représenter l'espace physique occupé par un objet de façon plus ou moins précise. Ces représentations se rapprochent des représentations utilisées dans les algorithmes de détection de collision. Plus de détails peuvent être trouvés dans un manuel de référence tel que Ericson (2005).

Seehof *et al.* (1966) et Xu *et al.* (2002) représentent les variables par des rectangles en deux dimensions. Chaque rectangle correspond au périmètre de la base d'un objet. Dans le cas des objets de forme non rectangulaire, le rectangle constitue un rectangle englobant, également à la base de l'objet. Baykan et Fox (1991) utilise également des rectangles, mais considère chaque arête comme une variable distincte.

Pfefferkorn (1975) et Charman (1993) représentent les variables par des polygones. Cette représentation offre plus de précision car elle permet d'être plus représentative de la véritable forme des objets à placer. La détermination du polygone consiste à déterminer l'enveloppe convexe de chaque objet.

Bonnefoi et Plemenos (1999), Le Roux et Gaildrat (2003) et Beldiceanu *et al.* (2007) utilisent une boîte en trois dimensions orientée selon le système de référence *(Axis Aligned Bounding Box, AABB*). Cette représentation a le désavantage d'être parfois très peu précise, lorsque les objets présentent une forme mince et longue. Toutefois, il est très facile de faire des tests de collision étant donné l'alignement avec les axes.

Akazawa *et al.* (2008) optent pour un peu plus de précision et utilisent des boîtes orientées selon les coordonnées de l'objet *(Object Oriented Box, OOB)*. Le volume englobant est évidemment plus précis que dans le cas des AABB, mais implique des calculs géométriques

un peu plus complexes.

Finalement, Shinya et Forgue (1995) et Tutenel *et al.* (2009a) utilisent un ensemble de polygones pour obtenir un volume englobant fidèle à la forme d'un modèle 3D. Le volume englobant est alors un volume convexe complexe très précis, mais implique une forte consommation de mémoire et des calculs plus coûteux.

La représentation des variables constitue donc un compromis entre une représentation physique fidèle des objets à placer et une complexité de résolution grandissante avec la fidélité.

3.4 Représentation des contraintes

Les variables de la planification spatiale étant les objets à placer, les contraintes permettent de définir des limitations par rapport aux possibilités de placement des objets. Les contraintes constituent donc des relations sur les objets à placer. Ces relations peuvent être unitaires, binaires ou impliquer un nombre variables d'objets. Cette section présente une revue des contraintes ayant été précédemment utilisées dans des systèmes de planification spatiale.

Il est à noter que la littérature par rapport à la planification spatiale présente généralement les contraintes impliquées dans les systèmes de façon intuitive. Cela signifie que l'implémentation de ces contraintes n'est habituellement pas présentée.

Étant le premier à formaliser le problème de planification spatiale, il est normal que Pfefferkorn (1975) ait également été le premier à définir plusieurs contraintes applicables à la planification spatiale. Il définit tout d'abord les relations binaires suivantes, permettant de placer un objet par rapport à un autre objet. Ces contraintes ont été globalement réutilisées dans les systèmes de planification spatiale subséquents.

Distance Une contrainte de distance indique la distance entre deux objets de façon qualitative ou quantitative.

Position Une contrainte de position indique une position relative d'un objet par rapport à un autre objet.

Orientation Une contrainte d'orientation indique une orientation relative d'un objet par rapport à un autre objet.

Adjacence Une relation d'adjacence indique qu'un objet doit être placé près d'un autre objet. Cette contrainte correspond à un cas particulier d'une contrainte de position.

En plus des contraintes de positionnement relatif, il définit des contraintes sur les perceptions et les mouvements que nous pouvons avoir avec les objets dans une scène. Ces contraintes constituent des concepts plus complexes, et n'ont pas encore été considérées dans

le contexte d'application des environnements 3D. La prise en compte de ces contraintes dans un système de planification spatiale pourrait aider à faciliter les animations dans les scènes générées.

Vue Une contrainte de vue indique qu'un objet devrait être visible à partir de la position d'un autre objet, et inversement.

Chemin Une contrainte de chemin indique qu'il doit exister un espace libre permettant de se déplacer d'un objet à un autre.

Finalement, il définit la contrainte d'espace, qui est la contrainte globale la plus importante à respecter pour la génération d'une scène réaliste. Cette contrainte est toujours considérée lors de la planification spatiale. La contrainte *geost* de Beldiceanu *et al.* (2007) est une implémentation de cette contrainte.

Espace Une contrainte d'espace indique qu'aucun des objets présents dans la scène ne doit occuper le même espace physique qu'un autre objet.

Baykan et Fox (1991) introduit une nouvelle contrainte de placement relative entre deux objets. Cette contrainte a la particularité de devoir exclure la paire d'objets de la contrainte d'espace.

Contenant Une contrainte de contenant indique que le volume correspondant à un objet doit être contenu à l'intérieur du volume d'un autre objet.

L'application de la planification spatiale à l'infographie permet d'introduire de nouvelles contraintes, en raison de la troisième dimension des environnements considérés. Xu *et al.* (2002) définissent les contraintes de surface et de support, qui sont de nouvelles contraintes de positionnement relatif entre deux objets.

Surface Une contrainte de surface indique la manière de placer un objet sur la surface d'un autre objet.

Support Une contrainte de support indique qu'un objet doit être supporté par un autre objet.

Pour la planification spatiale appliquée à l'infographie, la contrainte de support est la relation la plus importante à considérer. En effet, elle permet de s'assurer que les objets présents dans la scène soient placés sur un autre objet, comme c'est souvent le cas dans la réalité. Une bonne application de cette contrainte implique une combinaison avec une représentation sémantique de la scène. Ce concept est couvert en détail dans la section 3.6.

Le Roux et Gaildrat (2003) sont les premiers à appliquer une contrainte unitaire sur les objets, en considérant que les dimensions des objets à placer peuvent également être des variables soumises à des contraintes.

Dimension Une contrainte de dimension indique les valeurs ou limites possibles pour les dimensions d'un objet.

Ils complémentent également la contrainte de support de Xu *et al.* (2002) en définissant une contrainte de gravité. De cette manière, ils considèrent la gravité comme interne au CSP, plutôt qu'à l'aide d'une simulation physique externe.

Gravité Une contrainte de gravité invalide la position de tout objet ne reposant pas sur le sol ou un autre objet.

Finalement, ils introduisent le concept de contraintes de groupe. Ce type de contraintes s'applique à un ensemble d'objets similaires et permet de définir des configurations plus complexes.

Angulaire Une contrainte angulaire permet d'organiser un groupe d'objets en cercle autour d'un autre objet.

Akazawa *et al.* (2008) introduisent deux nouvelles contraintes de groupe. Ces deux types de contraintes facilitent la génération de scènes impliquant un grand nombre d'objets du même type. Il est à noter qu'ils étendent également la notion de support pour inclure les murs et le plafond comme supports possibles.

Regroupement Une contrainte de regroupement permet d'organiser un groupe d'objets comme un seul objet constitué de tous les objets du groupe connectés ensemble.

Espacement régulier Une contrainte d'espacement régulier permet d'organiser un groupe d'objet selon un espacement régulier dans une, deux ou trois dimensions.

Pour terminer, Tutenel *et al.* (2009a) introduisent une nouvelle contrainte unitaire. Cette contrainte se rapproche des contraintes de vue et de chemin, car elle se base sur l'interaction d'une personne avec la scène.

Dégagement Une contrainte de dégagement indique un espace autour d'un objet qui doit être laissé libre pour permettre son utilisation.

3.5 Représentation de la requête

La représentation de la requête permet de définir la scène qui devra être générée par la planification spatiale. Cette représentation doit contenir toutes les informations nécessaires pour produire la modélisation du problème sous forme de contraintes.

La requête peut être exprimée de trois manières différentes : par le biais d'un outil de modélisation 3D, d'un script ou d'une description textuelle.

3.5.1 Représentation dans un outil de modélisation

Les outils de modélisation permettent de manipuler les objets dans la scène. Le principal désavantage de ces outils est dû aux méthodes de contrôles limitatives offertes par les périphériques standards. Il est difficile de manipuler un objet possédant six degrés de liberté avec un périphérique ne possédant que deux degrés de liberté. Utiliser un outil de modélisation implique d'utiliser la planification spatiale pour assister la manipulation des objets dans l'environnement virtuel.

Le système pionnier *Sketchpad* (Sutherland (2003)) fut le premier système permettant la création interactive de scènes par l'entremise de contraintes. *Sketchpad* est un système de dessin assisté par ordinateur permettant de tracer, copier ou supprimer des figures à l'aide d'un crayon optique. Le système permet également l'application de contraintes géométriques sur les primitives de dessin qui seront assurées par le biais d'un système de satisfaction de contraintes.

Le système *ThingLab* (Borning (1981)) permet la construction d'environnements expérimentaux physiques ou géométriques reposant sur des systèmes de contraintes. Ces environnements incluent la simulation d'objets soumis à des contraintes géométriques. La satisfaction des contraintes repose sur un concept de hiérarchisation des contraintes, qui permet de soulager un problème n'ayant pas de solution, en tentant de minimiser les contraintes n'était pas satisfaites.

Shinya et Forgue (1995) utilisent un système de placement basé sur des manipulations. Le système définit six opérations de base permettant de déplacer les objets à l'intérieur d'une scène. Ces opérations permettent d'ajouter un objet à la scène, puis de le déplacer ou de le tourner selon plusieurs méthodes. L'utilisation de contraintes spatiales permet de restreindre les déplacements tentés par l'utilisateur en n'acceptant que des positions et orientations qui sont considérées valides pour chaque objet.

Xu *et al.* (2002) tentent de faciliter les manipulations en plaçant automatiquement les objets dans la scène. Chaque nouvel objet ajouté à la scène est automatiquement placé sur la surface d'un autre objet. Un nouvel objet est donc automatiquement placé sur le sol, ou de manière à reposer sur un autre objet. De plus, il est possible pour l'utilisateur de sélectionner la surface d'un objet. Le système pourra alors lui proposer d'autres objets qui sont généralement placés sur la surface choisie. L'attribution d'une position initiale repose sur l'utilisation d'une représentation des connaissances.

La combinaison de la planification spatiale avec un outil de manipulation permet certainement d'améliorer la méthode de configuration de scènes. Par contre, une telle représentation initiale ne permet pas de spécifier directement des contraintes particulières à appliquer sur les objets à placer. Les contraintes sont limitées à la logique présente dans la représentation

des connaissances.

3.5.2 Représentation sous forme de scripts

Les outils de placement automatique reposant sur une requête sous forme de scripts requièrent la lecture d'un fichier d'entrée qui décrit la scène à produire. Ces scripts contiennent généralement une description des objets à placer dans la scène, ainsi que les contraintes auxquelles ils doivent être soumis.

Bonnefoi et Plemenos (1999) proposent un système de requête sous forme de scripts basé sur l'expression de contraintes. Ces scripts permettent d'organiser la scène à produire selon une hiérarchie de sous-scènes. Ils permettent également de spécifier des contraintes entre plusieurs sous-scènes, ainsi que des contraintes entre des objets à placer.

Le Roux et Gaildrat (2003) proposent un format de script permettant de déclarer les objets à placer dans la scène et de définir des contraintes sur le placement de ces objets. Cependant, le script n'est qu'un moyen de tester le résolveur de contraintes, qui constitue l'aspect central du projet. Pour cette raison, le format de script utilisé présente un couplage fort avec le résolveur de contraintes.

Dans les deux cas, l'approche par script offre plus de flexibilité que l'intégration à un outil de modélisation. Par contre, les formats de scripts proposés sont fortement couplés avec la méthode de résolution, et ne présentent donc pas une abstraction permettant de décrire facilement la scène à représenter.

3.5.3 Représentation par le langage naturel

Certains projets de recherche se sont concentrés sur la création automatique de contenu multimédia selon une approche multimodale. Ces projets tentent de passer d'une représentation sous forme de texte à une représentation graphique.

Le système WordsEye, de Coyne et Sproat (2001), a pour but de générer automatiquement une scène 3D à partir d'une courte description textuelle. Le texte est parcouru et transformé en une représentation sémantique, à partir de laquelle les principaux éléments de la scène sont identifiés. Ces éléments permettent de choisir les objets 3D qui vont constituer la scène, de les positionner et de les manipuler pour obtenir la bonne pose.

Irawati *et al.* (2006) utilisent un système multimodal pour la manipulation d'objets dans un environnement virtuel. Le système utilise une grammaire dans le but de convertir les phrases parlées en texte. Ce principe demande de connaître à l'avance les mots que l'utilisateur peut utiliser. Chaque élément de la grammaire peut être converti en une règle définissant une manipulation d'objet.

Le système CONFUCIUS (Ma (2006)) vise à générer automatiquement des animations 3D à partir de texte. Ce système ne constitue pas réellement une application de la planification spatiale. En effet, le premier but de CONFUCIUS n'est pas de placer correctement des objets dans une scène, mais plutôt d'animer un modèle humanoïde pour représenter les actions associées à des verbes. Pour cette raison, l'analyse textuelle effectuée par CONFUCIUS est centrée sur l'analyse des verbes. CONFUCIUS propose une classification des verbes d'action dans le but de traduire leur représentation en animation.

Glass et Bangay (2007) visent à créer des scènes 3D et à en animer le contenu dans le temps. Leur système repose principalement sur des textes de fiction pour enfant. L'analyse du texte vise tout d'abord à identifier les espaces, les personnages et les objets inanimés. Par la suite, les relations spatiales et temporelles entre les objets sont identifiées. Le tout est organisé selon les références temporelles du récit, et les relations sont utilisées pour produire les trajectoires des objets dans le temps et dans l'espace.

Dans les trois cas, les ambiguïtés provenant de la langue naturelle sont présentées comme étant la difficulté principale à la réalisation d'un système de traduction de langage en graphiques. En effet, il est difficile d'obtenir une scène représentant exactement le texte en entrée, car le texte peut souvent être interprété de beaucoup de façons différentes.

3.6 Modèles sémantiques pour la planification spatiale

Cette section présente l'application d'un modèle sémantique à la planification spatiale. Nous commençons tout d'abord par définir le rôle et la nature d'une telle représentation. Par la suite, nous faisons une revue des modèles de représentation sémantique relatifs au placement d'objets dans une scène.

3.6.1 Motivation

L'un des principaux obstacles à la création automatique de scènes virtuelles est l'absence d'information contenue dans les ressources multimédia. En effet, cette information est limitée à l'aspect visuel des ressources. Pour cette raison, celles-ci ne contiennent pas d'information par rapport à leur nature ou à leur utilisation. Un modèle de représentation sémantique vise à combler ce manque d'information. Gutierrez *et al.* (2005) et Tutenel *et al.* (2008) défendent l'idée de coupler les ressources 3D avec de l'information sémantique.

Dans le cas de la planification spatiale, l'utilisation d'un modèle sémantique vise à aider à la prise de décision pour le positionnement des objets. En effet, pour attribuer une position à un objet dans un environnement virtuel, il est nécessaire de savoir à quoi sert cet objet, ainsi que de quel manière il est généralement disposé dans le monde réel. Pour cette raison,

Pfefferkorn (1975) et Baykan et Fox (1991) énoncent très tôt le besoin de connaissance comme faisant partie intégrale de la planification spatiale.

3.6.2 Définitions

Dans la littérature faisant référence à l'utilisation d'informations complémentaires pour aider au placement des objets, nous avons remarqué que les termes *sémantique* et *connaissance* étaient tous deux utilisés pour décrire ces types d'information. Nous tentons ici de définir l'application de ces deux termes au domaine de l'infographie dans le but d'éliminer l'ambiguïté pouvant être introduite par l'utilisation de deux termes.

Le terme *sémantique* provient de la notion de sens dans un langage. Dans un environnement virtuel, un objet sémantique est un objet qui inclut une définition de son propre sens. Par exemple, un modèle 3D sémantique pourrait définir sa classe, son rôle et son utilisation en plus de définir sa représentation visuelle.

Le terme *connaissance* regroupe un ensemble de définitions, de règles ou d'axiomes qu'un être humain peut connaître. En informatique, il est possible de regrouper toute cette information dans une base de connaissances. Dans le cadre de la planification spatiale une base de connaissances pourrait contenir des définitions d'objets et de relations pour définir comment placer les objets.

3.6.3 Modèles de représentation par scripts

L'une des premières tentatives d'élargir les informations dans les environnements virtuels est de coupler les ressources 3D avec des scripts. Kallmann et Thalmann (1998) présentent une structure qu'ils nomment *Smart Object*, consistant principalement en des objets décrivant différentes possibilités d'interaction dans un fichier de script.

La structure générale utilise le script comme descripteur de base de l'objet. Il peut définir différents niveaux de fonctionnalité, allant des propriétés intrinsèques de l'objet à des propriétés d'interactions avec des agents externes. Il est possible de réutiliser le même objet avec des scripts différents pour modéliser différents types d'interaction.

Le concept des *Smart Objects* offre de bons résultats. Cette approche est reprise et perfectionnée par Peters *et al.* (2003) et Abaci *et al.* (2005). Par contre, bien que cette approche constitue un bon moyen d'ajouter de l'information aux objets d'une scène 3D, elle ne permet pas facilement d'étendre la connaissance à plus d'un objet à la fois.

3.6.4 Modèles de représentation par bases de données

Afin d'avoir une forme de connaissance étendue à plus d'un objet à la fois, plusieurs systèmes ont tenté de modéliser la connaissance à l'intérieur d'une base de données communiquant avec une application.

Le principe de la base de données de Xu (2001) est d'assigner un type à chaque objet. Ces types sont classifiés selon une hiérarchie permettant d'indiquer des relations de supports entre les types. Chaque type possède donc une liste de types parents, sur lesquels il peut reposer, et une liste de types enfants, qui constituent des objets pour lesquels il peut jouer le rôle de support.

Les objets possèdent également des propriétés qui leur sont propres. Chaque objet possède des propriétés indiquant la nécessité ou l'impossibilité d'être placé sur un autre objet. De plus, chaque objet possède une orientation par défaut.

Tutenel *et al.* (2009b) utilisent également une base de données pour son générateur procédural de scènes d'intérieur. Par contre, cette fois-ci, la hiérarchie représentée est basée sur une relation d'héritage. De cette manière, chaque classe d'objet hérite des propriétés de son parent.

Les propriétés des objets correspondent à des espaces autour de ces objets. Ces propriétés peuvent indiquer des espaces ne pouvant être partagés par d'autres objets, des espaces nécessaires à l'utilisation d'un objet, ou tout autre espace pouvant avoir une utilisation particulière. Si on prend l'exemple d'un objet de type chaise, il n'est pas possible pour un autre objet d'occuper le même espace que la chaise et un espace est nécessaire devant la chaise pour permettre à une personne de venir s'y asseoir. De plus, l'espace sur le siège constitue un espace désigné pour permettre à une personne de s'asseoir et l'espace entre les pattes de la chaise peut être utilisé pour y ranger d'autres objets.

Cette base de données a pour but d'être utilisée conjointement avec des modèles 3D. Il est possible de définir un modèle 3D comme étant une instance d'une classe de la base de données. Pour cela, il est nécessaire d'indiquer les espaces autour du modèle correspondant aux diverses propriétés de la classe

L'utilisation d'une base de données constitue donc une approche beaucoup plus adaptée pour assister la planification spatiale. Le principal désavantage est le fort couplage qui est créé entre le code et la base de données. Dans les deux cas, chaque objet représenté dans la base de données est dupliqué comme une classe dans le code. Il est donc assez ardu d'étendre la base de données et de définir beaucoup de relations entre les objets.

3.6.5 Modèles de représentation par ontologies

Une base de données constitue un format assez approprié pour représenter la connaissance dans le but d'assister la planification spatiale. Par contre, cette structure offre des limitations par rapport à la représentation des contraintes ou des relations entre les concepts d'objets à placer.

En intelligence artificielle, le concept d'ontologie est justement utilisé pour représenter des concepts et des relations. Formellement, une ontologie définit un ensemble de concepts dans un domaine, ainsi qu'un ensemble de relations sur ces concepts. La structure d'une ontologie repose sur les langages de représentation tels que RDF et OWL.

Cette représentation peut être couplée avec un raisonneur pour effectuer des inférences par rapport à la structure de relations entre les concepts définis.

Une fois la structure de représentation d'une ontologie définie, il est possible de formuler des requêtes pour interroger l'ontologie. L'interrogation d'une ontologie représentée en OWL repose sur la structure du langage de requêtes SPARQL.

L'ontologie pourrait donc constituer une représentation de la connaissance très adaptée à la planification spatiale. Nous présentons tout d'abord une revue des applications de l'ontologie aux environnements multimédia. Par la suite, nous présentons quelques ontologies de domaines connexes qui pourraient être applicables à la planification spatiale.

Ontologies appliquées aux environnements virtuels

Otto (2005) utilise une ontologie pour proposer une méthode universelle de description des environnements virtuels. Selon sa représentation, chaque objet présent dans un environnement virtuel constitue un concept dans l'ontologie. Ces concepts sont organisés selon un système de ressources, où chaque ressource peut être constituée de plusieurs formes géométriques. Le système de relations est utilisé pour appliquer des transformations géométriques à chacun des objets, selon une structure similaire à un graphe de scène. Cette représentation est intéressante car il s'agit d'une des premières applications de l'ontologie aux environnements 3D. Par contre, la représentation est assez limitée. Elle tente d'appliquer la représentation de graphe de scène à une ontologie, plutôt que d'appliquer la sémantique et la logique d'une ontologie à l'infographie.

Irawati *et al.* (2006) proposent une ontologie axée sur la sélection d'éléments dans une scène 3D. La particularité de cette solution est de diviser l'ontologie en deux niveaux : un niveau général et un niveau spécifique à l'application développée. L'ontologie modélise des objets dans l'espace, et les classifie en objets statiques et en objets déplaçables. Les relations entre les objets sont utilisées pour indiquer les placements permis pour chaque objet. Ces

relations consistent à indiquer les objets sur lesquels un autre objet peut être placé. Cette ontologie constitue une approche intéressante, mais la frontière entre le niveau général et le niveau spécifique est difficile à évaluer, et reste tout de même très fortement couplée avec le domaine d'application.

Kessing *et al.* (2010) utilisent une approche ontologique pour représenter les objets. En plus des classes et des propriétés habituelles, ils introduisent les notions de service et d'action. Un service constitue une définition de la fonction associée à un objet. Par exemple, un manteau fournit le service de réchauffer. Pour bénéficier de ce service, une action doit être préalablement réalisée, il est nécessaire de porter le manteau. Cette représentation constitue une application très intéressante de l'ontologie aux environnements virtuels. Par contre, les informations représentées se rapprochent plutôt des *Smart Objects* et modélisent les interactions possibles avec des objets plutôt que la manière de les placer dans l'espace.

Ontologies appliquées au traitement de la langue

Le traitement de la langue implique d'être capable de reconnaître la syntaxe et la sémantique d'un texte. Pour cette raison, ce domaine utilise des ressources regroupant les concepts présents dans une langue. Ces ressources constituent une représentation qui pourrait s'appliquer à la planification spatiale.

WordNet (Miller (1995)) est une ressource organisant les mots de la langue anglaise. Elle regroupe plus de 118000 mots et plus de 90000 sens. Ces mots sont organisés selon des classes de verbes, d'adjectifs, de noms, et d'adverbes. Ils sont reliés entre eux par six types de relations, les deux principales étant la synonymie et l'antonymie. La totalité du réseau sémantique créé par WordNet est accessible globalement à travers un fureteur web.

Similairement à WordNet, FrameNet (Baker *et al.* (1998)) est une ressource lexicographique disponible sur internet. Cette ressource regroupe chaque verbe selon chacun de ses sens selon toutes les possibilités sémantiques et syntaxiques. La construction de FrameNet a été basée sur l'annotation manuelle à partir de corpus de référence. FrameNet contient actuellement plus de 11600 unités lexicales et plus de 960 *frames*.

Ces deux ressources constituent une excellente organisation des concepts présents dans une langue, et une telle organisation pourrait être très bénéfique pour la planification spatiale. Par contre, dans de nombreux cas, ces organisations manquent de précision par rapport aux paramètres visuels ou spatiaux des concepts, et s'appliquent donc mal à l'infographie ou à la planification spatiale.

Ontologies pour la représentation du sens commun

La planification spatiale implique de connaître la nature, l'utilisation et la configuration habituelle des objets à placer dans la scène. Une telle connaissance se rapproche de ce qu'on appelle le *sens commun* par rapport à la représentation de la connaissance. Le sens commun regroupe toute la connaissance qui est considérée comme évidente pour la plupart des êtres humains, mais qui doit être modélisée pour être utilisable par un ordinateur.

Cyc (Lenat (1995)) est la plus grande ressource de sens commun développée à ce jour. Elle présente 10^5 concepts reliés par 10^6 axiomes. Chaque assertion de Cyc est située dans un ou plusieurs contextes. Une assertion définie dans un contexte indique que ce concept est toujours vrai dans le contexte considéré. Pour éviter les conflits entre assertions, celles-ci sont ordonnées selon leur probabilité de se produire dans la réalité. De cette manière, il est possible de raisonner sur la véracité d'une proposition sans reposer sur des valeurs numériques supposées. Le principal obstacle à l'utilisation de Cyc est sa grande envergure. En effet, l'utilisation d'une si grande base de connaissances demande un investissement de temps initial remarquable pour se familiariser avec sa structure et ses concepts.

Pour cette raison, ConceptNet (Liu et Singh (2004)) tente de représenter le même type de relations que Cyc, mais en utilisant une structure de réseaux sémantiques similaire à celle de WordNet. ConceptNet contient plus de 250000 éléments de sens commun, et 19 types de relations pouvant relier ces éléments.

Bien que ces ressources modélisent une connaissance très proche des besoins de la planification spatiale, elles sont peu adaptées au placement d'objets dans l'espace. Ainsi, de nombreuses relations qui seraient nécessaires pour configurer les objets correctement sont manquantes. Particulièrement, la notion de support, qui est considérée comme la relation la plus importante entre les objets, n'est présente dans aucune des deux représentations.

Ontologies pour la représentation de l'espace

L'un des principaux obstacles à la conversion de texte en graphique repose sur la difficulté d'exprimer le résultat de l'analyse de texte dans l'espace. En effet, les technologies de traitement de la langue permettent d'identifier la structure et le sens des phrases. Par contre, la visualisation du sens d'une phrase dans l'espace est faite intuitivement par l'être humain, et n'est généralement incluse dans la phrase elle-même.

Bateman *et al.* (2010) propose *GUM-Space*, une extension de l'ontologie générale *GUM*, spécialisée pour la représentation de l'espace pour le traitement de la langue. Le type de base pouvant être décrit par *GUM* est une configuration. Une configuration représente une situation dans le temps et dans l'espace. Le constituant principal d'une configuration est le

processus, qui correspond à une action. Selon l'action faite, différents paramètres peuvent être spécifiés, pour décrire potentiellement les éléments impliqués dans cette action, ainsi que les paramètres spatiaux entre ces éléments. Tous les paramètres de *GUM-Space* sont structurés selon une hiérarchie de classes spécifiques. Ces classes permettent de définir de manière très précise et non ambiguë les actions et modalités spatiales.

La représentation de *GUM-Space* constitue une excellente représentation des concepts de relations spatiales. Cette représentation serait donc un bon moyen de désambiguïser l'ensemble des concepts spatiaux provenant de l'analyse de texte dans le cadre d'une application de traduction de texte en animation.

3.6.6 Modèles de représentation par apprentissage machine

Les données provenant d'utilisateurs peuvent constituer une méthode très efficace pour emmagasiner une structure de connaissance. En effet, les actions faites par les utilisateurs d'un système reflètent exactement les données que l'ajout de connaissances souhaite combler.

En plus de sa base de connaissances, Xu (2001) utilise également une approche d'apprentissage. En effet, toutes les tentatives de placement des utilisateurs sont sauvegardées sous la forme de scénarios. Ces scénarios peuvent alors être utilisés pour modifier le poids associé à certaines relations, ou bien à déterminer les orientations ou positions par défaut des objets.

Les travaux de Jeff Orkin et de son *Restaurant Game* (Orkin (2007)) constituent une excellente façon de recueillir de l'information par apprentissage. À travers une simulation sur internet, toutes les données d'interactions ont été recueillies entre deux utilisateurs, l'un jouant le rôle du serveur, l'autre le rôle du client. De cette manière un réseau de connaissances très détaillé a pu être obtenu au niveau des interactions à l'intérieur d'un restaurant.

3.7 Résumé critique

En résumé, les techniques de planification spatiale constituent une approche tout à fait appropriée pour l'automatisation de la configuration d'objets dans un environnement virtuel. En représentant les objets à placer comme les variables d'un problème de satisfaction de contraintes, il est possible d'utiliser la programmation par contraintes pour obtenir une configuration valide de scène respectant un ensemble de contraintes définies. Par contre, il existe tout de même des éléments de cette approche pouvant être l'objet de nouvelles améliorations.

3.7.1 Amélioration des formats d'entrée

Dans un premier temps, nous avons soulevé le fait que les formats d'entrée pour la planification spatiale sont à améliorer. La représentation à travers un outil de modélisation ne

permet pas de formuler des contraintes spécifiques à une scène en particulier. La représentation par script est assez flexible mais repose sur une structure mathématique spécifique et ne présente pas une abstraction suffisante pour représenter une scène facilement. Finalement, la représentation textuelle permet à la fois de décrire des scènes variées et un haut niveau d'abstraction, mais présente des ambiguïtés issues du langage qui sont difficiles à traiter.

Il serait intéressant de pouvoir utiliser une représentation possédant un haut niveau d'abstraction, et permettant de décrire facilement des scènes variées pouvant contenir des contraintes spécifiques, sans le niveau d'ambiguïté présent avec l'utilisation du langage naturel.

3.7.2 Amélioration des modèles sémantiques

Une planification spatiale efficace nécessite une combinaison avec un modèle de représentation sémantique. Similairement aux formats de représentation initiale, les modèles de représentation sémantique utilisés précédemment apportent peu de flexibilité quant aux scènes pouvant être modélisées, et sont généralement très difficiles à construire et à maintenir.

Il serait intéressant de pouvoir utiliser un format de représentation de la connaissance adapté au problème de planification spatiale. Le format d'une ontologie semble être un excellent candidat à cette représentation. Par contre, il n'existe actuellement aucune ontologie développée dans l'optique d'une utilisation conjointe avec la planification spatiale.

3.7.3 Traitement des rotations d'objets dans la méthode de résolution

La programmation par contraintes demeure la méthode de résolution la plus efficace pour le problème de planification spatiale. Cependant, la plupart des systèmes utilisés à ce jour supportent difficilement la rotation des objets dans les configurations de scènes produites. Pour que la planification spatiale constitue une automatisation complète du positionnement d'objets dans une scène virtuelle, il est important que cette solution incorpore les six degrés de liberté pouvant être considérés pour un objet à placer. Pour cette raison, il serait intéressant d'apporter un effort additionnel aux méthodes de résolution pour y incorporer la rotation des objets.

3.7.4 Implémentation des contraintes de perception

La revue de littérature nous a permis de présenter un grand nombre de contraintes pouvant s'appliquer sur les objets à placer dans un environnement virtuel. Par contre, il n'est pas clair que la totalité des contraintes décrites dans cette section aient déjà été implémentées dans le cadre d'un système de planification spatiale. Plus particulièrement, certaines contraintes ont

été définies dans le domaine d'application du placement architectural, mais ces contraintes n'ont pas encore été considérées dans le domaine d'application de l'infographie.

Les contraintes de perception constituent un ensemble de contraintes qui seraient particulièrement intéressantes à incorporer dans un système de planification spatiale pour l'infographie. En effet, ces contraintes permettraient de s'assurer qu'il est possible de voir un objet à partir d'un autre objet, de pouvoir circuler entre les objets de la scène, et que l'espace nécessaire à l'utilisation d'un objet est correctement représenté dans la scène.

3.7.5 Définition de métriques d'évaluation

La programmation par contraintes est présentée comme étant une méthode de résolution efficace pour la planification spatiale, et plusieurs systèmes affirment avoir pu implémenter un système de planification spatiale avec succès. Pourtant, il n'existe pratiquement aucune métrique d'évaluation appliquée à la planification spatiale. Il n'est donc pas possible d'évaluer la qualité d'un système de planification spatiale ou de comparer une approche avec une autre.

Il serait intéressant de proposer des méthodes d'évaluation appliquées à la planification spatiale. De telles métriques permettraient non seulement de comparer les approches et méthodes de résolution, mais faciliterait également l'analyse des résultats de chaque méthode.

CHAPITRE 4

SOLUTION PROPOSÉE

Ce chapitre présente la solution que nous proposons pour améliorer la planification spatiale. Le but de notre solution est d'offrir une abstraction de haut niveau combinée avec une représentation de la connaissance sous la forme d'une ontologie pour faciliter la modélisation de la planification spatiale.

La présentation de notre solution est organisée en quatre sections différentes. Nous présentons tout d'abord notre représentation abstraite de scène. Puis, nous présentons la structure de notre représentation de la connaissance pour la planification spatiale. Ensuite, nous présentons la façon dont nous avons modélisé le CSP à partir de la représentation abstraite de scène et de l'ontologie. Finalement, nous présentons notre implémentation pour la traduction de la représentation abstraite en système de contraintes.

4.1 Représentation abstraite de scène

Notre représentation abstraite repose sur une adaptation de l'ontologie *GUM-Space* pour la définition d'un format XML décrivant une scène à modéliser. Nous décrivons dans un premier temps l'abstraction ayant été faite par rapport aux représentations existantes, puis nous présentons les détails du format de représentation abstraite proposé.

4.1.1 Abstraction des informations dans le format d'entrée

La première étape pour la définition d'une représentation abstraite de scène consiste à identifier la quantité d'information minimale nécessaire à la description d'une scène virtuelle. Cette information minimale constitue l'abstraction sur laquelle repose notre représentation.

Ainsi, l'abstraction que nous proposons repose sur la représentation des informations suivantes pour la définition d'une scène à générer par planification spatiale :

- Une liste des objets à placer
- Une liste des relations entre les objets à placer

Cette abstraction correspond aux variables et contraintes utilisées pour la modélisation du CSP de la planification spatiale. Cependant, nous sommes intéressés à représenter ces mêmes concepts, mais de façon à y inclure le minimum d'informations possible. Nous obtenons donc une abstraction en retirant l'ensemble des attributs numériques que ces éléments devraient contenir dans la formulation d'un CSP.

Ainsi, chaque objet à placer ne correspond pas à une aire ou à un volume dans l'espace. Il s'agit seulement d'un identifiant associé au concept de cet objet. Similairement, les relations décrites dans cette représentation ne constituent pas des contraintes. Il ne s'agit que d'un identifiant associé au concept de la relation. Chaque relation ne comporte donc aucun attribut numérique, mais plutôt des caractérisations qualitatives par rapport au placement des objets.

L'abstraction de notre représentation nous permet donc de modéliser la scène comme un ensemble de concepts représentant des objets et un ensemble de concepts représentant des relations appliquées sur ces objets. La traduction de cette représentation en une modélisation résoluble par programmation par contraintes implique donc d'introduire les éléments numériques nécessaires à la résolution.

4.1.2 Structure de *GUM-Space*

L'ontologie *GUM-Space*, présentée dans la section 3.6.5, formalise la définition des relations spatiales dans le cadre de l'analyse de texte. Tel que soulevé lors de la revue de littérature, cette formalisation constitue un excellent moyen de résoudre l'ambiguïté associée à la langue, tout en présentant une excellente abstraction pour la description d'une scène virtuelle.

La structure de *GUM-Space* repose sur le concept de configuration. Une configuration correspond à une situation dans laquelle une action a lieu. À l'intérieur de chaque configuration, il est possible de spécifier des objets et des relations pour indiquer les aspects spatiaux de l'action en cours. Cette structure définit une hiérarchie complexe de types d'actions et de types de relations entre les objets. Toutes ces relations sont classifiées selon l'impact qu'elles ont sur les objets dans l'espace.

Prenons l'exemple d'une phrase analysée selon le formalisme de *GUM-Space* pour illustrer la structure des configurations. La figure 4.1 présente l'annotation de la phrase *A hat is on the rack.* selon les principes d'étiquetage de *GUM-Space*.

```
SpatialLocating 'sl1'
    locatum SimpleThing 'hat' A hat
    processInConfiguration Process 'being' is
    placement GeneralizedLocation 'gl1'
        relatum SimpleThing 'rack' the rack
        hasSpatialModality Support 'sup1' on
```

Figure 4.1 Annotation d'une phrase de type SpatialLocating

La première information identifiée est le type de configuration dont il est question. Dans le cas de cette phrase-ci, il s'agit d'une configuration de type *SpatialLocating*. Cela nous indique que l'on veut décrire le placement d'un objet dans l'espace, et qu'aucune action ne se déroule dans cette configuration. Par la suite, on décrit une entité de type *locatum*. Cet élément doit toujours être présent dans une configuration de type *SpatialLocating*, car il s'agit de l'objet dont on décrit le placement. Il est caractérisé par l'appellation *SimpleThing* pour indiquer qu'il s'agit d'un objet. La troisième information présentée est le *process*. Dans le cas de notre exemple, il n'y a pas d'action en cours, le *process* est donc indiqué par le verbe *being*, qui décrit plutôt un état. Finalement, on identifie un élément de type *placement*. C'est ce qui permet de décrire la façon dont le *locatum* est placé. Dans notre cas, le placement est indiqué par une *GeneralizedLocation*, ce qui indique un lieu dans l'espace. Ce *placement* est caractérisé par un *relatum* et une *spatialModality*. Le *relatum* décrit l'objet par rapport auquel le *locatum* est positionné, et la *spatialModality* indique la façon dont cet objet est placé. Il a donc bien été identifié que le chapeau (*locatum*) est placé par rapport au rack (*relatum*), qui lui sert de support(*spatialModality*).

Prenons un second exemple pour illustrer la façon dont le formalisme varie d'une configuration à l'autre, et profitons de cet exemple pour présenter quelques concepts plus avancés de *GUM-Space*. La figure 4.2 est une annotation de la phrase *He puts the ball in the box*.

```
AffectingDirectedMotion 'adm1'
     actor SimpleThing 'he' He
     processInConfiguration Process 'putting' puts
     actee SimpleThing 'ball' the ball
     route GeneralizedRoute 'gr1'
          destination GeneralizedLocation 'gl1'
               relatum SimpleThing 'box' the box
               hasSpatialModality Containment 'con1' in
```

Figure 4.2 Annotation d'une phrase de type AffectingDirectedMotion

Nous retrouvons dans cette analyse quelques éléments également présents dans l'exemple précédent. Nous avons à nouveau un *process*, il s'agit cette fois-ci du verbe *putting*, qui indique l'action en cours. Nous avons également des éléments de type *SimpleThing* qui constituent les objets présents dans la phrase. Par contre, nous avons cette fois-ci une configuration de type *AffectingDirectedMotion*. Ce type de configuration décrit une action définie par une orientation, et qui a un effet sur un objet. L'action décrite permet effectivement de déplacer la

balle dans une direction pour la placer dans la boite. De plus, nous pouvons voir que les objets sont caractérisés par les éléments *actor* et *actee*. L'*actor* définit l'objet qui effectue l'action, alors que l'*actee* définit l'objet qui est affecté par l'action. Finalement, cette configuration contient un objet de type *GeneralizedRoute*. Ce type d'élément permet de définir un chemin dans l'espace. Dans cet exemple, ce chemin est défini par un lieu à l'intérieur de la boite. Nous remarquons que la route ne définit pas de source, puisque la phrase ne spécifie pas l'emplacement initial de la balle.

Ces deux exemples nous ont permis d'illustrer quelques éléments de la structure définie par l'ontologie *GUM-Space*. Ce formalisme est extrêmement bien adapté au type d'abstraction que nous voulons représenter, en raison de sa grande précision par rapport aux façons de caractériser l'espace.

4.1.3 Adaptation de *GUM-Space* pour la création d'un format de représentation abstraite

Afin de pouvoir utiliser *GUM-Space* comme représentation abstraite, nous avons adapté la structure de l'ontologie selon un format XML que nous utilisons pour la formulation de nos requêtes. Ainsi, l'exemple de la figure 4.1 suivra la structure XML adaptée de la figure 4.3.

```
<sceneDeclaration>
  <configuration xsi:type="SpatialLocating">
    <process>fn:Being_located</process>
    <locatum>
        <name>Hat1</name>
        <class>ph:Hat</class>
     </locatum>
    <location>
      <relatum>
        <name>Rack1</name>
         <class>ph:Rack</class>
      </relatum>
      <spatialModality>
        <type>gum:Support</type>
      </spatialModality>
    </location>
  </configuration>
</sceneDeclaration>
```

Figure 4.3 Représentation d'une phrase de type SpatialLocating

La structure de la requête suit donc clairement la structure proposée de *GUM-Space*. Nous retrouvons la présence d'une configuration de type *SpatialLocating* et un *process* correspondant au verbe être. Nous retrouvons également les éléments *locatum*, *relatum* et *spatialModality* indiquant respectivement le chapeau, le rack et la notion de support. La notion

de *placement* est remplacé par un élément de type *location*, qui est également permis par une configuration de type *SpatialLocating*.

Similairement, l'exemple de la figure 4.2 peut être transformé selon notre format pour atteindre la structure de la figure 4.4.

```
<sceneDeclaration>
  <configuration xsi:type="AffectingDirectedMotion">
    <process>fn:Placing</process>
    <actor>
      <name>He</name>
      <class>ph:Male</class>
    </actor>
    <actee>
      <name>Ball1</name>
      <class>ph:Ball</class>
    </actee>
    <route xsi:type="GeneralizedRoute">
      <destination xsi:type="GeneralizedLocation">
        <relatum>
          <name>Box1</name>
          <class>ph:Box</class>
        </relatum>
        <spatialModality>
          <type>gum:Containment</type>
        </spatialModality>
      </destination>
    </route>
  </configuration>
</sceneDeclaration>
```

Figure 4.4 Représentation d'une phrase de type AffectingDirectedMotion

Nous voyons encore une fois que les éléments présents dans l'annotation sont organisés similairement selon la structure XML. Nous retrouvons une configuration de type *Affecting-DirectedMotion*. Les concepts d'*actor* et d'*actee* permettent d'identifier l'individu ainsi que la balle. Finalement, nous avons une *GeneralizedRoute* introduisant une destination définie comme étant l'intérieur de la boite.

En plus des éléments qui étaient également présents dans la structure des annotations, notre format introduit quelques notions supplémentaires. La principale est l'apparition des préfixes *fn*, *ph* et *gum*. Ces préfixes constituent des liens vers des ontologies spécifiques. Le préfixe *fn* fait référence à l'ontologie *framenet* et nous permet de classifier l'action en cours. C'est la raison pour laquelle le *process* est différent dans le second exemple, car nous l'avons relié au frame *Placing* dans l'ontologie *framenet*. Similairement, chaque objet est caractérisé par un élément *class*, qui nous permet de faire un lien vers notre propre base de connaissances, caractérisée par le préfixe *ph*. Finalement, le préfixe *gum* fait référence à l'ontologie *GUM-*

Space, et nous permet ainsi de garder un lien direct vers les *spatialModality* impliquées.

Cette structure nous permet donc d'encapsuler dans des éléments XML les informations de notre abstraction de scène. Les objets sont caractérisés par un nom et une classe, et les relations entre ces objets sont indiquées par des *spatialModality* s'il y a lieu. Cette représentation nous permet donc de représenter aisément une scène virtuelle à générer automatiquement à partir du concept de configurations introduit dans *GUM-Space*. La définition de scènes complexes consiste à introduire plusieurs configurations à l'intérieur d'un même élément de type *sceneDeclaration*.

Cette représentation a également l'avantage de supporter la représentation d'informations pouvant s'étendre à plus grand que la génération de scènes par planification spatiale. Le fait que *GUM-Space* soit défini dans le cadre de l'analyse de texte pourrait aider à traduire du texte en animation. De plus, le fait que les configurations supportent les concepts d'actions, cette représentation abstraite peut également être utilisée pour la création d'animations à l'intérieur de la scène générée.

4.2 Représentation de la connaissance

Notre représentation de la connaissance consiste en une ontologie que nous avons modélisée parallèlement à notre expérimentation dans le but d'évaluer la faisabilité d'utiliser une structure ontologique pour la planification spatiale à partir d'une représentation abstraite.

Cette représentation de la connaissance constitue un moyen de modéliser le sens commun associé au placement spatial. Nous définissons dans un premier temps une hiérarchie de classes dans le but de modéliser les concepts d'objets à positionner. Par la suite, nous définissons des relations entre ces concepts, dans le but de représenter les règles de placement qui sont généralement utilisés dans le monde.

Il est à noter que ces règles de placement ne sont pas absolues, et qu'il serait possible de prioriser des relations de la représentation abstraite lors de conflit entre la représentation abstraite et la représentation de la connaissance. La connaissance a donc comme but principal de combler l'absence d'information plutôt que d'être un guide de placement rigide.

4.2.1 Hiérarchie de classes

La structure de classe de notre ontologie nous permet de représenter l'ensemble des concepts considérés dans notre représentation de la connaissance. Ces concepts regroupent évidemment les objets à placer, mais s'étendent également à des concepts pouvant assister au placement des objets et à la définition de relations de placement. Notre hiérarchie de classes nous permet donc de définir des objets, des contrôleurs ainsi que des actions.

Hiérarchie d'objets

Les classes d'objets correspondent à des objets pouvant être placés dans un environnement virtuel. La figure 4.5 montre l'ensemble de notre hiérarchie d'objets.

Les objets sont organisés selon une arborescence de classes dans laquelle les feuilles constituent des objets concrets, alors que les autres classes constituent des types abstraits. La particularité des types concrets est qu'ils sont les seuls à représenter des objets visant à être placés dans une scène.

L'ensemble des classes d'objets peuvent être caractérisées par des propriétés ou des relations. Une propriété ou une relation s'appliquant sur un objet concret permettra de caractériser cet objet s'il est ajouté dans une scène. Une propriété ou une relation appliquée à un objet abstrait s'appliquera à tout objet concret dérivé lorsqu'il est ajouté dans une scène. Pour cette raison, il est préférable de caractériser les objets au niveau le plus abstrait possible, car cela permet d'affecter le plus d'objets possibles en limitant le nombre de liens dans l'ontologie.

Prenons l'exemple de la classification de mobilier pour illustrer la différence entre les types concrets et abstraits. La figure 4.6 montre la hiérarchie des objets de type *Furniture*. Dans cette hiérarchie, *Chair*, *Sofa*, *Bed*, *Shelf* et *Table* sont des types concrets, alors que *Furniture*, *SleepingFurniture*, *StoringFurniture* et *WorkingFurniture* sont des types abstraits.

À l'intérieur de cette architecture, nous définissons que tout objet de type *Furniture* doit reposer sur le sol. Ainsi, tous les objets concrets dérivant de *Furniture* devront reposer sur le sol s'ils sont ajoutés à une scène. Nous définissons également qu'un objet de type *SittingFurniture* peut permettre à une personne de s'asseoir. Une personne pourrait donc s'asseoir sur un sofa, une chaise ou un lit, mais pas sur une étagère. De plus, nous définissons qu'un objet de type *SleepingFurniture* peut permettre à une personne de dormir. Il serait donc possible d'utiliser un lit pour dormir, mais pas une chaise.

Cette hiérarchie permet donc d'organiser l'ensemble des objets pouvant être présents dans la scène. Elle permet également d'utiliser des références à certains types abstraits pour faciliter l'expression de propriétés ou de relations.

Propriétés numériques des objets

Chaque objet concret représenté dans l'ontologie possède des propriétés numériques. Ces propriétés regroupent la masse et les dimensions d'une classe d'objets, et sont représentées par un intervalle numérique. L'utilisation d'un intervalle nous permet d'obtenir un certain niveau de réalisme en évitant que tous les objets du même type soient identiques.

La figure 4.7 montre les propriétés numériques de la classe *Refrigerator*. Les intervalles

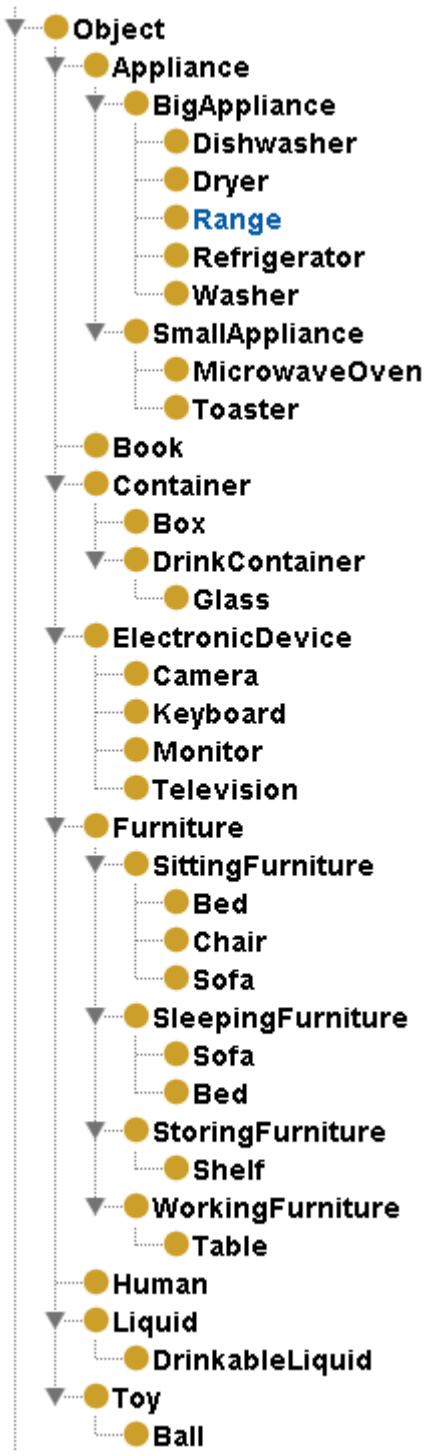

Figure 4.5 Hiérarchie de classes d'objets de la base de connaissances

de masse et de dimensions y sont clairement définis. Ainsi, plusieurs réfrigérateurs dans une même scène pourront avoir des dimensions différentes, tout en respectant ces intervalles de valeurs.

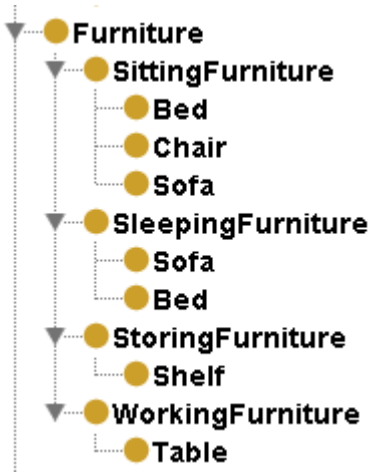

Figure 4.6 Hiérarchie de classes pour les objets de type Furniture

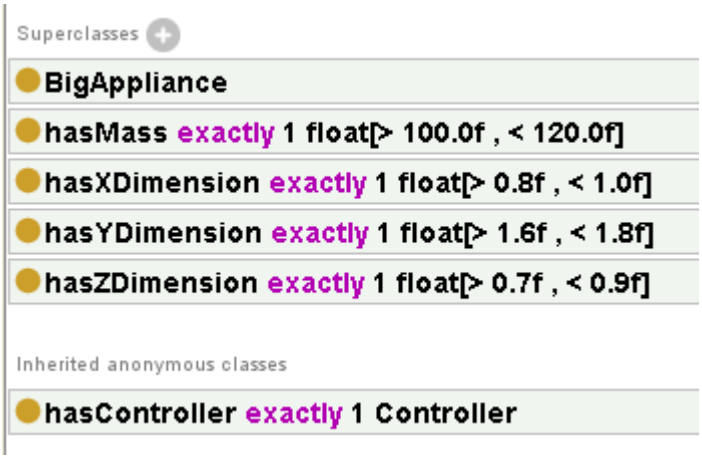

Figure 4.7 Propriétés numériques de la classe Refrigerator

Hiérarchie de contrôleurs

Tous les objets présents dans une scène doivent être assignés à un contrôleur. Le concept de *contrôleur* représente un élément de la scène qui affecte un objet de manière à lui permettre de rester immobile. Concrètement, cela signifie qu'un objet n'est pas affecté par la gravité en raison de l'action d'un contrôleur. Notre hiérarchie de contrôleurs est visible à la figure 4.8.

Cette hiérarchie définit plusieurs types abstraits de contrôleurs. Ces contrôleurs affectent les objets de la manière suivante :

Container Un contrôleur de type *container* permet de définir un objet comme étant à l'intérieur d'un autre objet. Par exemple, un réfrigérateur agit comme *container* pour un litre de lait situé à l'intérieur du réfrigérateur.

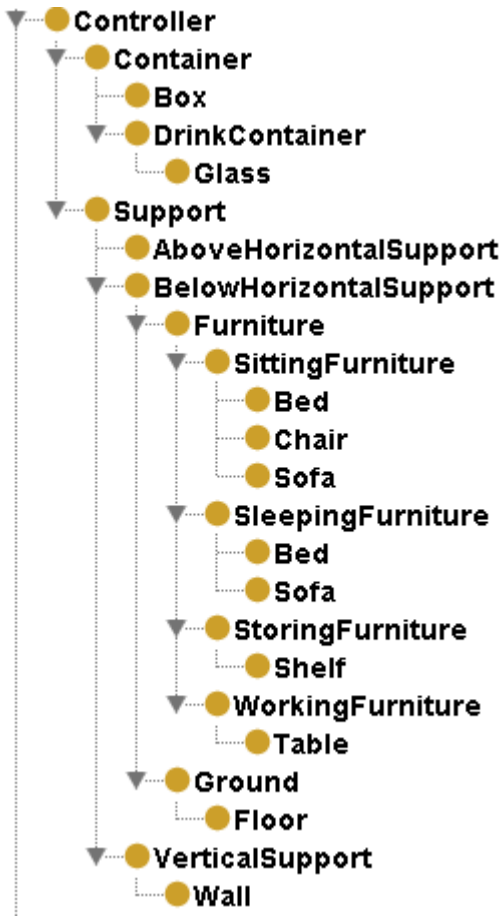

Figure 4.8 Hiérarchie de classes de contrôleurs de la base de connaissances

Support Le contrôleur de type *support* correspond à la relation de support habituelle de la planification spatiale. Nous avons divisé ce concept en trois types de support distincts.

AboveHorizontalSupport Un support de type *AboveHorizontalSupport* définit un support d'orientation horizontale s'appliquant au-dessus de l'objet. Par exemple, le plafond agit comme support pour un ventilateur plafonnier.

BelowHorizontalSupport Un support de type *BelowHorizontalSupport* définit un support d'orientation horizontale s'appliquant en-dessous de l'objet. Ce type correspond à la notion de support habituelle. Par exemple, un bureau sur lequel repose un cahier agit comme support sur ce cahier.

VerticalSupport Un support de type *VerticalSupport* définit un support d'orientation verticale. Par exemple, un mur agit comme support pour un tableau qui y serait accroché.

Nous pouvons voir une redondance partielle par rapport à la hiérarchie d'objets. Cela signifie qu'il existe des types d'objets, abstraits comme concrets, qui sont également des

contrôleurs. Cependant, la hiérarchie de contrôleurs est présentée comme distincte de la hiérarchie d'objets car il existe des contrôleurs qui ne constituent pas des objets. En effet, le sol, ainsi que le plancher, les murs ou le plafond d'une pièce ne constituent pas des objets à placer, mais jouent le rôle de support sur ces objets. Il est cependant possible pour un concept d'être à la fois un contrôleur et un objet à placer. Ainsi, une table peut à la fois être un objet à placer dans la scène et jouer le rôle de support pour un verre, qui lui, joue le rôle de contenant pour de l'eau.

Le choix de ne pas considérer le sol, le plancher et les murs vise à simplifier la génération de scène dans notre système. En effet, nous considérons que les objets à placer sont tous situés dans la même pièce. Pour la génération de scènes impliquant plusieurs pièces, il serait nécessaire d'attribuer une pièce à chaque objet à placer, et à effectuer une résolution de planification spatiale pour chaque pièce considérée.

Classes d'actions

Les actions, représentées par le type *process* dans l'ontologie, constituent la dernière hiérarchie de classes modélisée. Celle-ci est présentée à la figure 4.9.

Figure 4.9 Hiérarchie de classes d'actions de la base de connaissances

Les actions ne constituent en aucun cas des objets à placer dans la scène, et ne constituent pas un concept habituellement considéré pour la planification spatiale. Celles-ci sont toutefois considérées dans notre représentation de la connaissance, puisque nous pouvons les représenter dans notre représentation abstraite. En effet, elles correspondent au type *process* qui est également défini dans la représentation abstraite.

Les actions peuvent être considérées dans les décisions de placement faites à partir de la base de connaissances. Puisque nous connaissons l'action que nous avons à représenter, il nous est possible d'utiliser la base de connaissances pour nous assurer que la scène produite contient les éléments nécessaires à la création d'une animation dans la scène générée.

Prenons comme exemple le cas de l'action de manger. Pour être capable de manger, il est nécessaire que la scène à générer contienne de la nourriture. De plus, selon le type de

nourriture dont il est question, nous aurons besoin d'ajouter à la scène un récipient pour contenir cette nourriture, et un ou plusieurs ustensiles qui seront utilisés pour manger cette nourriture. L'inclusion de types d'action dans notre représentation de la connaissance nous permet donc d'assister la génération de scènes par l'ajout automatique d'objets nécessaires à la réalisation de cette action.

4.2.2 Relations

L'intérêt d'utiliser une ontologie provient principalement de la possibilité de définir des relations sur l'ensemble des concepts définis par l'ontologie. Ces relations définissent des axiomes connus par rapport à certaines classes d'objets. La figure 4.10 contient l'ensemble des relations que nous avons définies.

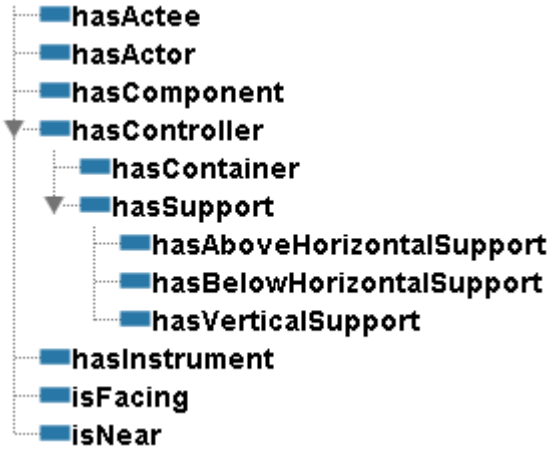

Figure 4.10 Relations de la base de connaissances

Il nous est possible d'interroger l'ontologie par rapport à ces relations dans le but d'assister notre planification spatiale. Chaque relation peut être utilisée pour accomplir un ou plusieurs des buts suivants.

Dans un premier temps, certaines relations peuvent être utilisées pour ajouter des objets à ceux définis par la requête de scène définie dans la représentation abstraite. L'ajout de ces objets peut être dû à la nécessité de placer un objet pour en placer un autre, ou bien pour augmenter le nombre d'objets dans la scène afin d'augmenter son niveau de réalisme.

Dans un second temps, certaines relations visent à définir la façon de placer un objet par rapport à un autre objet. Ainsi, une fois que nous savons que deux objets sont présents dans la scène, il est possible d'utiliser ce type de relation pour ajouter des contraintes quant au placement d'un objet par rapport à un autre.

Certaines relations peuvent servir à accomplir les deux buts cités ci-haut. Pour cette raison, nous avons plutôt regroupé les relations modélisées selon le type d'objets qu'elles impliquent, selon les hiérarchies définies précédemment.

Relations impliquant des contrôleurs

Notre ontologie définit les relations suivantes impliquant un objet et un contrôleur.

- hasContainer
- hasAboveHorizontalSupport
- hasBelowHorizontalSupport
- hasVerticalSupport

Chacune de ces relations permet de spécifier le contrôleur nécessaire au positionnement de chaque objet considéré dans l'ontologie. Ces relations ont pour objectif d'être capable d'ajouter le contrôleur à chacun des objets contenus dans la requête, mais également d'appliquer une contrainte de support ou de contenant entre l'objet et le contrôleur. Dans le cas où l'objet de la relation est un type abstrait, le contrôleur à ajouter à la scène doit être un sous-type concret dérivé de l'objet de la relation.

Relations impliquant des actions

Notre ontologie définit les relations suivantes impliquant un objet et une action.

- hasActor
- hasActee
- hasInstrument

Dans les trois cas, ces relations servent à s'assurer que les objets nécessaires à la réalisation d'une action soient présents dans la scène. Ces relations peuvent donc être utilisées pour ajouter de nouveaux objets à la scène, mais n'offrent pas d'indication par rapport à leur placement.

Il aurait été possible de regrouper ces trois relations en une seule relation. Cependant, la distinction pourrait être utile pour la création automatique d'animations à l'intérieur des scènes qui auront été générées.

Relations impliquant seulement des objets

Finalement, les relations suivantes ne s'appliquent qu'entre les types de la hiérarchie d'objets.

- isNear
- isFacing

Ces relations permettent d'indiquer qu'un objet est généralement placé près d'un autre objet, ou qu'un objet fait face à un autre objet. Ces relations peuvent donc servir à l'application de contraintes de position ou d'orientation. Il est également possible d'ajouter des objets à la représentation abstraite à partir de ces relations.

4.3 Représentation des variables

Dans le cadre de notre solution, nous avons choisi d'utiliser la représentation de type *Object Oriented Bounding Box* pour les variables de notre problème. Ainsi, chaque objet est défini par une position dans l'espace et par une dimension selon chacune des trois dimensions de l'espace.

4.3.1 Position

La position de l'objet dans l'espace correspond à un vecteur à trois dimensions. Cette propriété est modélisée par trois variables entières ayant pour domaine les dimensions de la scène. Chaque variable correspond à la position de l'objet selon les axes directeurs du système de coordonnées. La position de l'objet est fixée au centre du rectangle à la base de l'objet.

4.3.2 Dimensions

Tableau 4.1 Domaines de dimensions des objets dans la base de connaissances

Object	x	y	z
Bed	[1.90, 2.20]	[0.30, 0.60]	[1.50, 1.70]
Book	[0.12, 0.20]	[0.03, 0.08]	[0.15, 0.30]
Chair	[0.55, 0.60]	[0.80, 1.00]	[0.44, 0.60]
Dishwasher	[0.60, 0.70]	[0.85, 0.87]	[0.60, 0.62]
Dryer	[0.75, 0.80]	[0.95, 1.00]	[0.65, 0.70]
MicrowaveOven	[0.30, 0.50]	[0.25, 0.30]	[0.45, 0.55]
Sofa	[0.85, 0.95]	[0.80, 0.90]	[1.40, 2.80]
Table	[0.75, 1.05]	[0.70, 0.80]	[0.75, 2.93]
Toaster	[0.20, 0.35]	[0.25, 0.30]	[0.30, 0.55]
Refrigerator	[0.80, 1.00]	[1.60, 1.80]	[0.70, 0.90]
Washer	[0.80, 1.00]	[0.75, 1.00]	[0.60, 0.65]

Les dimensions de l'objet sont également modélisées par trois valeurs entières. Cependant, contrairement aux valeurs de la position, les dimensions de l'objet sont des valeurs constantes, et non des variables pour la résolution du CSP.

Les dimensions d'un objet sont choisies à l'intérieur d'un domaine défini comme un intervalle de valeurs à l'intérieur de l'ontologie. La table 4.1 présente les domaines de valeurs associés à chaque objet de l'ontologie.

4.4 Représentation des contraintes

Le but de notre projet étant d'évaluer la possibilité d'utiliser une représentation abstraite pour la planification spatiale, nous avons limité les relations considérées à un petit ensemble. Cette section présente tout d'abord les relations de façon intuitive, puis présente les équations définissant chaque relation sous la forme de contraintes.

Notre système considère les contraintes suivantes :

- Proximal
- Distal
- FrontProjectionExternal
- BackProjectionExternal
- RightProjectionExternal
- LeftProjectionExternal
- GroundSupport
- Support
- Space

Les relations de type *FrontProjectionExternal*, *BackProjectionExternal*, *LeftProjectionExternal* et *RightProjectionExternal* correspondent aux contraintes de positions décrites dans la section 3.4. Nos relations de positionnement relatif se basent sur les coordonnées de référence de la scène pour distinguer la gauche de la droite et l'avant de l'arrière.

Les relations de type *Proximal* et *Distal* correspondent aux contraintes de distance. Ces relations permettent de contraindre une paire d'objets à être près ou loin l'un de l'autre. La limite de distance définissant la frontière entre le près et le loin varie selon la dimension des objets impliqués.

En plus de ces contraintes, nous traitons également la contrainte de support, puisque celle-ci constitue la relation la plus importante pour la planification spatiale. La relation de type *GroundSupport* indique qu'un objet repose sur le sol. La relation de type *Support* indique qu'un objet repose sur un autre objet. Ces relations de support correspond à la relation *BelowHorizontalSupport* de notre ontologie. Les relations de type *AboveHorizontalSupport* et *VerticalSupport* n'ont pas été considérées.

Finalement, nous considérons la contrainte d'espace, selon laquelle deux objets ne peuvent occuper le même espace physique. L'utilisation de cette contrainte repose sur l'implémenta-

tion de la contrainte globale *geost* Beldiceanu *et al.* (2007) dans le résolveur de contraintes *Choco* Jussien, N. and Rochart, G. and Lorca (2008) .

4.4.1 LeftProjectionExternal

La relation *LeftProjectionExternal* contraint le *locatum* à être placé à la gauche du *relatum*.

$$zDistance = \frac{locatum.dimensions.z + relatum.dimensions.z}{2} \tag{4.1}$$

$$locatum.position.z <= relatum.position.z - zDistance \tag{4.2}$$

4.4.2 RightProjectionExternal

La relation *RightProjectionExternal* contraint le *locatum* à être placé à la droite du *relatum*.

$$zDistance = \frac{locatum.dimensions.z + relatum.dimensions.z}{2} \tag{4.3}$$

$$locatum.position.z >= relatum.position.z + zDistance \tag{4.4}$$

4.4.3 FrontProjectionExternal

La relation *FrontProjectionExternal* contraint le *locatum* à être placé à l'avant du *relatum*.

$$xDistance = \frac{locatum.dimensions.x + relatum.dimensions.x}{2} \tag{4.5}$$

$$locatum.position.x <= relatum.position.x - xDistance \tag{4.6}$$

4.4.4 BackProjectionExternal

La relation *RightProjectionExternal* contraint le *locatum* à être placé à l'arrière du *relatum*.

$$xDistance = \frac{locatum.dimensions.x + relatum.dimensions.x}{2} \tag{4.7}$$

$$locatum.position.x >= relatum.position.x + xDistance \tag{4.8}$$

4.4.5 Proximal

La relation *Proximal* contraint le *locatum* à être placé à une distance du *relatum* qui est inférieure à la somme de leurs dimensions selon les axes horizontaux.

$$xDistance = 2 \times (locatum.dimensions.x + relatum.dimensions.x) \tag{4.9}$$

$$zDistance = 2 \times (locatum.dimensions.z + relatum.dimensions.z) \tag{4.10}$$

$$|locatum.position.x - relatum.position.x| < xDistance \tag{4.11}$$

$$|locatum.position.z - relatum.position.z| < zDistance \tag{4.12}$$

4.4.6 Distal

La relation *Distal* contraint le *locatum* à être placé à une distance du *relatum* qui est supérieure à la somme de leurs dimensions selon les axes horizontaux.

$$xDistance = 2 \times (locatum.dimensions.x + relatum.dimensions.x) \tag{4.13}$$

$$zDistance = 2 \times (locatum.dimensions.z + relatum.dimensions.z) \tag{4.14}$$

$$|locatum.position.x - relatum.position.x| > xDistance \tag{4.15}$$

$$|locatum.position.z - relatum.position.z| > zDistance \tag{4.16}$$

4.4.7 GroundSupport

La relation *GroundSupport* constraint le *locatum* à reposer sur le sol.

$$locatum.position.y = 0 \tag{4.17}$$

4.4.8 Support

La relation *GroundSupport* constraint le *locatum* à reposer sur le *relatum*.

$$xDistance = \frac{relatum.dimensions.x}{2} \tag{4.18}$$

$$zDistance = \frac{relatum.dimensions.z}{2} \tag{4.19}$$

$$locatum.position.x < relatum.position.x - xDistance \tag{4.20}$$

$$locatum.position.x > relatum.position.x + xDistance \tag{4.21}$$

$$locatum.position.z < relatum.position.z - zDistance \tag{4.22}$$

$$locatum.position.z > relatum.position.z + zDistance \tag{4.23}$$

$$locatum.position.y = relatum.position.y + relatum.dimensions.y \tag{4.24}$$

4.5 Traduction de la requête en CSP

La dernière étape de notre solution implique de réaliser la génération de scène par planification spatiale à partir de la représentation abstraite en utilisant notre modélisation du problème. Cette génération implique plusieurs étapes de traitement visant à convertir la représentation de la requête en un modèle de CSP, puis à obtenir une solution en résolvant ce CSP.

Cette section présente les différentes étapes de conversion par le biais d'un exemple détaillé. La description de chaque étape est accompagnée des données correspondantes pour l'exemple considéré.

4.5.1 Représentation abstraite

À partir des objets présents dans l'ontologie, il est facile de créer une requête décrivant une scène dans laquelle sont placés des objets se retrouvant généralement dans un appartement. L'exemple considéré est une scène impliquant 13 objets de l'ontologie que nous avons soumis à 13 contraintes.

Commençons par définir un premier groupe d'objets, correspondant à la cuisine de l'appartement. Les deux configurations de la figure 4.5.1 nous permettent d'ajouter un réfrigérateur, un four et une table à notre scène. Dans un premier temps, nous spécifions que le réfrigérateur doit se trouver à droite du four, alors que la table doit se trouver à sa gauche. De plus, nous spécifions que les trois objets doivent être près les uns des autres.

```
<configuration xsi:type="SpatialLocating">
  <process>fn:Being_located</process>
  <locatum>
      <name>Refrigerator1</name>
      <class>ph:Refrigerator</class>
   </locatum>
  <location>
    <relatum>
      <name>Range1</name>
       <class>ph:Range</class>
    </relatum>
    <spatialModality>
      <type>gum:RightProjectionExternal</type>
    </spatialModality>
    <spatialModality>
      <type>gum:Proximal</type>
    </spatialModality>
  </location>
</configuration>
```

```
<configuration xsi:type="SpatialLocating">
  <process>fn:Being_located</process>
  <locatum>
      <name>Table1</name>
      <class>ph:Table</class>
   </locatum>
  <location>
    <relatum>
      <name>Range1</name>
       <class>ph:Range</class>
    </relatum>
    <spatialModality>
      <type>gum:LeftProjectionExternal</type>
    </spatialModality>
    <spatialModality>
      <type>gum:Proximal</type>
    </spatialModality>
  </location>
</configuration>
```

Figure 4.11 Configurations pour la disposition d'un lit, d'un four et d'une table

Nous pouvons compléter ce groupe d'objets en utilisant la table en tant que support. Ainsi, les configurations de la figure 4.5.1 nous permettent d'ajouter un grille-pain ainsi qu'un four à micro-ondes de façon à ce qu'ils reposent sur la table.

```
<configuration xsi:type="SpatialLocating">
  <process>fn:Being_located</process>
  <locatum>
      <name>Toaster1</name>
      <class>ph:Toaster</class>
   </locatum>
  <location>
    <relatum>
      <name>Table1</name>
       <class>ph:Table</class>
    </relatum>
    <spatialModality>
      <type>gum:Support</type>
    </spatialModality>
  </location>
</configuration>
```

```
<configuration xsi:type="SpatialLocating">
  <process>fn:Being_located</process>
  <locatum>
      <name>Microwave1</name>
      <class>ph:MicrowaveOven</class>
   </locatum>
  <location>
    <relatum>
      <name>Table1</name>
       <class>ph:Table</class>
    </relatum>
    <spatialModality>
      <type>gum:Support</type>
    </spatialModality>
  </location>
</configuration>
```

Figure 4.12 Configurations pour la disposition du grille-pain et d'un four à micro-ondes sur la table

Nous définissons un deuxième groupe d'objets se situant loin de notre premier groupe. Nous commençons par ajouter un lit et une table, en indiquant introduisant une relation de type *Distal* entre le lit et le four. Ce positionnement est représenté par les configurations de la figure 4.5.1.

```
<configuration xsi:type="SpatialLocating">
  <process>fn:Being_located</process>
  <locatum>
      <name>Bed1</name>
      <class>ph:Bed</class>
   </locatum>
  <location>
    <relatum>
      <name>Table2</name>
      <class>ph:Table</class>
    </relatum>
    <spatialModality>
      <type>gum:Proximal</type>
    </spatialModality>
  </location>
</configuration>
```

```
<configuration xsi:type="SpatialLocating">
  <process>fn:Being_located</process>
  <locatum>
      <name>Bed1</name>
      <class>ph:Bed</class>
   </locatum>
  <location>
    <relatum>
      <name>Range1</name>
      <class>ph:Range</class>
    </relatum>
    <spatialModality>
      <type>gum:Distal</type>
    </spatialModality>
  </location>
</configuration>
```

Figure 4.13 Configurations pour la disponition d'un lit et d'une table

Nous utilisons à nouveau la table comme support. Cependant, plutôt que de placer plu-

sieurs objets directement sur la table, nous ajoutons trois livres que nous plaçons de façon à former une pile sur la table. Ainsi, deux des livres vont avoir un livre comme support plutôt que la table. Les configurations de la figure 4.5.1 nous permettent de définir une pile de trois livres reposant sur la table.

```
<configuration xsi:type="SpatialLocating">
  <process>fn:Being_located</process>
  <locatum>
      <name>Book1</name>
      <class>ph:Book</class>
   </locatum>
  <location>
    <relatum>
      <name>Table2</name>
      <class>ph:Table</class>
    </relatum>
    <spatialModality>
      <type>gum:Support</type>
    </spatialModality>
  </location>
</configuration>
```

```
<configuration xsi:type="SpatialLocating">
  <process>fn:Being_located</process>
  <locatum>
      <name>Book2</name>
      <class>ph:Book</class>
   </locatum>
  <location>
    <relatum>
      <name>Book1</name>
      <class>ph:Book</class>
    </relatum>
    <spatialModality>
      <type>gum:Support</type>
    </spatialModality>
  </location>
</configuration>
```

```
<configuration xsi:type="SpatialLocating">
  <process>fn:Being_located</process>
  <locatum>
      <name>Book3</name>
      <class>ph:Book</class>
   </locatum>
  <location>
    <relatum>
      <name>Book2</name>
      <class>ph:Book</class>
    </relatum>
    <spatialModality>
      <type>gum:Support</type>
    </spatialModality>
  </location>
</configuration>
```

Figure 4.14 Configurations pour la disposition de trois livres sur une table.

Nous terminons notre scène en ajoutant quelques autres objets. Nous ajouter un dernier objet à notre second groupe, en définissant une chaise devant être située à proximité de la table. Finalement, nous définissons un dernier groupe d'objets contenant une machine à laver et une machine à sécher les vêtements. Ces trois nouveaux objets sont ajoutés à la scène par le biais des configurations de la figure 4.5.1.

```
<configuration xsi:type="SpatialLocating">
  <process>fn:Being_located</process>
  <locatum>
      <name>Chair1</name>
      <class>ph:Chair</class>
   </locatum>
  <location>
    <relatum>
      <name>Table2</name>
      <class>ph:Table</class>
    </relatum>
    <spatialModality>
      <type>gum:Proximal</type>
    </spatialModality>
  </location>
</configuration>
```

```
<configuration xsi:type="SpatialLocating">
  <process>fn:Being_located</process>
  <locatum>
      <name>Dryer1</name>
      <class>ph:Dryer</class>
   </locatum>
  <location>
    <relatum>
      <name>Washer1</name>
      <class>ph:Washer</class>
    </relatum>
    <spatialModality>
      <type>gum:Proximal</type>
    </spatialModality>
  </location>
</configuration>
```

Figure 4.15 Configurations pour la disposition d'une chaise et de machines à laver

4.5.2 Attribution de dimensions aux objets de la scène

L'initialisation des variables implique de déterminer le domaine de valeurs à attribuer à chaque variable. Tel que mentionné dans la section 4.3.1, le domaine de chaque variable positionnelle correspond aux dimensions de la scène. Cependant, les dimensions de la scène dépend également des objets qu'elle contient et de leurs dimensions. L'initialisation des variables demande donc de déterminer les dimensions des objets considérés et les dimensions de la scène de façon à déterminer le domaine des variables.

L'attribution de dimensions aux objets se fait à partir des données de l'ontologie. Pour chaque objet de la scène, la dimension selon chaque axe est choisie à l'intérieur des intervalles de dimensions définis dans l'ontologie. La table 4.2 présente les dimensions qui ont été attribuées aux objets de l'exemple.

Dans un premier temps, nous pouvons remarquer que les dimensions attribuées ont été converties en centimètre, afin d'ajouter de la précision à la résolution, puisque nous utilisons des valeurs entières. Nous pouvons également remarquer que les dimensions correspondent aux domaines présentés à la table 4.1. Finalement, nous pouvons voir que les objets de la même classe ne possèdent des dimensions différentes les uns des autres.

4.5.3 Initialisation de la scène

Une fois que nous avons déterminé les dimensions des objets de la scène, nous pouvons déterminer les dimensions de la scène. La scène est initialisée avec les plus petites dimensions permettant de contenir tous les objets considérés, peu importe les contraintes appliquées. Les dimensions de la scène sont donc calculées selon les équations suivantes :

Tableau 4.2 Dimension des objets de l'exemple détaillé

Object	Class	x	y	z
Bed1	Bed	199.0	40.0	159.0
Book1	Book	13.0	7.0	25.0
Book2	Book	14.0	3.0	18.0
Book3	Book	15.0	3.0	27.0
Chair1	Chair	57.0	85.0	47.0
Dryer1	Dryer	78.0	96.0	65.0
Microwave1	MicrowaveOven	48.0	27.0	54.0
Range1	Range	68.0	115.0	71.0
Refrigerator1	Refrigerator	82.0	167.0	89.0
Table1	Table	103.0	72.0	271.0
Table2	Table	103.0	72.0	127.0
Toaster1	Toaster	32.0	28.0	48.0
Washer1	Washer	92.0	97.0	61.0

$$scene.dimensions.x = \sum_{i=1}^{n}(4 \times object_i.dimensions.x) \tag{4.25}$$

$$scene.dimensions.y = \sum_{i=1}^{n}(object_i.dimensions.y) \tag{4.26}$$

$$scene.dimensions.z = \sum_{i=1}^{n}(4 \times object_i.dimensions.z) \tag{4.27}$$

Selon les axes horizontaux x et z, les dimensions de la scène permettent donc de traiter le pire cas, celui de la relation *Distal*, selon laquelle les objets peuvent nécessiter une distance égale au double de leur dimensions. Selon l'axe vertical y, les dimensions de la scène permettent de placer la totalité des objets les uns sur les autres.

Dans le cadre de notre exemple, nous obtenons donc une scène avec les dimensions suivantes :

$$scene.dimensions.x = 1808 \tag{4.28}$$

$$scene.dimensions.y = 812 \tag{4.29}$$

$$scene.dimensions.z = 2137 \tag{4.30}$$

4.5.4 Initialisation du domaine des variables

À partir des dimensions de la scène et de chacun des objets, nous pouvons déterminer le domaine des variables du CSP. Ce domaine correspond aux dimensions de la scène, que nous ajustons en fonction des dimensions des objets de façon à leur permettre d'être placés entièrement dans la scène. Ces domaines sont calculés à partir des équations suivantes :

$$x_{min} = \frac{objet.dimensions.x}{2} \tag{4.31}$$

$$x_{max} = scene.dimensions.x - x_{min} \tag{4.32}$$

$$dom_x = [x_{min}, x_{max}] \tag{4.33}$$

$$y_{min} = 0 \tag{4.34}$$

$$y_{max} = scene.dimensions.y - object.dimensions.y \tag{4.35}$$

$$dom_y = [y_{min}, y_{max}] \tag{4.36}$$

$$z_{min} = \frac{objet.dimensions.z}{2} \tag{4.37}$$

$$z_{max} = scene.dimensions.z - z_{min} \tag{4.38}$$

$$dom_z = [z_{min}, z_{max}] \tag{4.39}$$

4.5.5 Modèle Choco : Variables

L'initialisation des variables décrite par les équations des sections précédentes nous permet de lire la représentation abstraite du problème et de passer ces variables au résolveur *Choco*. Nous obtenons la modélisation suivante en ce qui concerne les variables du problème.

$$Bed1.position.x \in [99, 3517] \tag{4.40}$$

$$Bed1.position.y \in [0, 772] \tag{4.41}$$

$$Bed1.position.z \in [79, 4182] \tag{4.42}$$

$$Book1.position.x \in [6, 3610] \tag{4.43}$$

$$Book1.position.y \in [0, 805] \tag{4.44}$$

$$Book1.position.z \in [12, 4249] \tag{4.45}$$

$$Book2.position.x \in [7, 3609] \tag{4.46}$$

$$Book2.position.y \in [0, 809] \tag{4.47}$$

$$Book2.position.z \in [9, 4252] \tag{4.48}$$

$$Book3.position.x \in [7, 3609] \tag{4.49}$$

$$Book3.position.y \in [0, 809] \tag{4.50}$$

$$Book3.position.z \in [13, 4248] \tag{4.51}$$

$$Chair1.position.x \in [28, 3588] \tag{4.52}$$

$$Chair1.position.y \in [0, 727] \tag{4.53}$$

$$Chair1.position.z \in [23, 4238] \tag{4.54}$$

$$Dryer1.position.x \in [39, 3577] \tag{4.55}$$

$$Dryer1.position.y \in [0, 716] \tag{4.56}$$

$$Dryer1.position.z \in [32, 4229] \tag{4.57}$$

$$Microwave1.position.x \in [24, 3592] \tag{4.58}$$

$$Microwave1.position.y \in [0, 785] \tag{4.59}$$

$$Microwave1.position.z \in [27, 4234] \tag{4.60}$$

$$Range1.position.x \in [34, 3582] \tag{4.61}$$

$$Range1.position.y \in [0, 697] \tag{4.62}$$

$$Range1.position.z \in [35, 4226] \tag{4.63}$$

$$Refrigerator1.position.x \in [41, 3575] \tag{4.64}$$

$$Refrigerator1.position.y \in [0, 645] \tag{4.65}$$

$$Refrigerator1.position.z \in [44, 4217] \tag{4.66}$$

$$Table1.position.x \in [51, 3565] \tag{4.67}$$

$$Table1.position.y \in [0, 740] \tag{4.68}$$

$$Table1.position.z \in [135, 4126] \tag{4.69}$$

$$Table2.position.x \in [51, 3565] \tag{4.70}$$

$$Table2.position.y \in [0, 740] \tag{4.71}$$

$$Table2.position.z \in [63, 4198] \tag{4.72}$$

$$Toaster1.position.x \in [16, 3600] \tag{4.73}$$

$$Toaster1.position.y \in [0, 784] \tag{4.74}$$

$$Toaster1.position.z \in [24, 4237] \tag{4.75}$$

$$Washer1.position.x \in [46, 3570] \tag{4.76}$$

$$Washer1.position.y \in [0, 715] \tag{4.77}$$

$$Washer1.position.z \in [30, 4231] \tag{4.78}$$

Il est assez facile de recalculer l'attribution de ces domaines à partir des valeurs et des équations des sections 4.5.2, 4.5.3 et 4.5.4.

4.5.6 Modèle Choco : Contraintes

L'autre étape de la résolution du problème implique d'ajouter les contraintes au modèle Choco afin de pouvoir attribuer une valeur à chaque variable à l'intérieur de son domaine. Nous présentons ici les équations résolues de l'exemple de façon à montrer la correspondance entre la représentation abstraite, le modèle CSP, la solution obtenue et les équations de la section 4.4. Ces équations impliquent les variables correspondant à la position des objets, des valeurs constantes correspondant aux dimensions des objets, ainsi que des valeurs temporaires, correspondant à certaines valeurs intermédiaires définies dans les équations des contraintes.

Commençons tout d'abord par les équations les plus simples de la résolution, les équations relatives à l'axe vertical. Nous avons dans un premier temps les équations correspondant à la relation de *GroundSupport*, impliquant les objets reposant sur le sol.

$$Bed1.position.y : 0 == 0 \tag{4.79}$$

$$Chair1.position.y : 0 == 0 \tag{4.80}$$

$$Dryer1.position.y : 0 == 0 \tag{4.81}$$

$$Range1.position.y : 0 == 0 \tag{4.82}$$

$$Table1.position.y : 0 == 0 \tag{4.83}$$

$$Table2.position.y : 0 == 0 \tag{4.84}$$

$$Refrigerator1.position.y : 0 == 0 \tag{4.85}$$

$$Washer1.position.y : 0 == 0 \tag{4.86}$$

Nous avons par la suite les équations correspondant à la relation de type *Support*. Ces équations ont permis de déterminer la position verticale des objets reposant sur un autre objet à partir de la position et de la dimension verticale du support. Nous avons tout d'abord les équations permettant de placer le grille-pain et le four à micro-ondes sur la première table, puis les équations permettant de placer les trois livres sur la seconde table.

$$Microwave1.position.y : 72 = Table1.position.y : 0 + 72 \tag{4.87}$$

$$Toaster1.position.y : 72 = Table1.position.y : 0 + 72 \tag{4.88}$$

$$Book1.position.y : 72 = Table2.position.y : 0 + 72 \tag{4.89}$$

$$Book2.position.y : 79 = Book1.position.y : 72 + 7 \tag{4.90}$$

$$Book3.position.y : 82 = Book2.position.y : 79 + 3 \tag{4.91}$$

Si nous continuons avec la relation de *Support*, nous nous souvenons que la relation indique que le centre de l'objet supporté doit se trouver à l'intérieur de la projection verticale du rectangle défini par les dimensions horizontale du support. La position horizontale des objets supportés a été déterminée par les équations suivantes :

$$Book3.position.x : 858 >= Book2.position.x : 854 - 6 \tag{4.92}$$

$$Book3.position.z : 298 >= Book2.position.z : 305 - 8 \tag{4.93}$$

$$Book2.position.x : 854 >= Book3.position.x : 858 - 6 \tag{4.94}$$

$$Book2.position.z : 305 >= Book3.position.z : 298 - 8 \tag{4.95}$$

$$Book1.position.x : 855 >= Book2.position.x : 854 - 5 \tag{4.96}$$

$$Book2.position.z : 305 >= Book1.position.z : 302 - 11 \tag{4.97}$$

$$Book1.position.z : 302 >= Book2.position.z : 305 - 11 \tag{4.98}$$

$$Book2.position.x : 854 >= Book1.position.x : 855 - 5 \tag{4.99}$$

$$Book1.position.x : 855 >= Table2.position.x : 828 - 50 \tag{4.100}$$

$$Book1.position.z : 302 >= Table2.position.z : 299 - 62 \tag{4.101}$$

$$Table2.position.x : 828 >= Book1.position.x : 855 - 50 \tag{4.102}$$

$$Table2.position.z : 299 >= Book1.position.z : 302 - 62 \tag{4.103}$$

$$Table1.position.x : 1487 >= Toaster1.position.x : 1451 - 50 \tag{4.104}$$

$$Table1.position.z : 1691 >= Toaster1.position.z : 1689 - 134 \tag{4.105}$$

$$Toaster1.position.x : 1451 >= Table1.position.x : 1487 - 50 \tag{4.106}$$

$$Toaster1.position.z : 1689 >= Table1.position.z : 1691 - 134 \tag{4.107}$$

$$Microwave1.position.x : 1480 >= Table1.position.x : 1487 - 50 \tag{4.108}$$

$$Table1.position.x : 1487 >= Microwave1.position.x : 1480 - 50 \tag{4.109}$$

$$Microwave1.position.z : 1588 >= Table1.position.z : 1691 - 134 \tag{4.110}$$

$$Table1.position.z : 1691 >= Microwave1.position.z : 1588 - 134 \tag{4.111}$$

Dans le cadre de notre exemple, les relations de projection *RightProjectionExternal* et *LeftProjectionExternal* ont été appliquées au réfrigérateur et à la première table dans le but de la placer de part et d'autre du four. Ces relations sont données par les équations résolues suivantes :

$$CSTE_{320} : 320 >= |TMP_{22} : -95| + 1 \tag{4.112}$$

$$CSTE_{300} : 300 >= |TMP_{20} : -213| + 1 \tag{4.113}$$

$$1 \times TMP_{22} : -95 + 1 \times Refrigerator1.position.z : 2058 + -1 \times Range1.position.z : 1963 = 0 \tag{4.114}$$

$$1 \times TMP_{20} : -213 + 1 \times Refrigerator1.position.x : 1705 + -1 \times Range1.position.x : 1492 = 0 \tag{4.115}$$

La relation *Proximal* a été appliquée à plusieurs paires d'objets dans le but de nous assurer que ces objets seraient près les uns des autres dans la scène générée. L'application de cette relation correspond aux équations résolues suivantes :

$$CSTE_{684} : 684 >= |TMP_{18} : 272| + 1 \tag{4.116}$$

$$1 \times TMP_{18} : 272 + 1 \times Table1.position.z : 1691 + -1 \times Range1.position.z : 1963 = 0 \tag{4.117}$$

$$CSTE_{342} : 342 >= |TMP_{16} : 5| + 1 \tag{4.118}$$

$$1 \times TMP_{16} : 5 + 1 \times Table1.position.x : 1487 + -1 \times Range1.position.x : 1492 = 0 \tag{4.119}$$

$$CSTE_{320} : 320 >= |TMP_{4} : -265| + 1 \tag{4.120}$$

$$1 \times TMP_4 : -265 + 1 \times Chair1.position.x : 1093 + -1 \times Table2.position.x : 828 = 0 \quad (4.121)$$

$$CSTE_{348} : 348 >= |TMP_6 : 9| + 1 \quad (4.122)$$

$$1 \times TMP_6 : 9 + 1 \times Chair1.position.z : 290 + -1 \times Table2.position.z : 299 = 0 \quad (4.123)$$

$$CSTE_{604} : 604 >= |TMP_8 : 156| + 1 \quad (4.124)$$

$$1 \times TMP_8 : 156 + 1 \times Bed1.position.x : 672 + -1 \times Table2.position.x : 828 = 0 \quad (4.125)$$

$$CSTE_{572} : 572 >= |TMP_{10} : -347| + 1 \quad (4.126)$$

$$1 \times TMP_{10} : -347 + 1 \times Bed1.position.z : 646 + -1 \times Table2.position.z : 299 = 0 \quad (4.127)$$

$$CSTE_{252} : 252 >= |TMP_2 : 116| + 1 \quad (4.128)$$

$$1 \times TMP_2 : 116 + 1 \times Dryer1.position.z : 81 + -1 \times Washer1.position.z : 197 = 0 \quad (4.129)$$

$$CSTE_{340} : 340 >= |TMP_0 : -184| + 1 \quad (4.130)$$

$$1 \times TMP_0 : -184 + 1 \times Dryer1.position.x : 325 + -1 \times Washer1.position.x : 141 = 0 \quad (4.131)$$

Finalement, la relation *Distal* appliquée entre le lit et le four est appliquée par les équations résolues suivantes :

$$|TMP_{12} : 820| >= CSTE_{534} : 534 + 1 \quad (4.132)$$

$$1 \times TMP_{12} : 820 + 1 \times Bed1.position.x : 672 + -1 \times Range1.position.x : 1492 = 0 \quad (4.133)$$

$$|TMP_{14} : 1317| >= CSTE_{460} : 460 + 1 \quad (4.134)$$

$$1 \times TMP_{14} : 1317 + 1 \times Bed1.position.z : 646 + -1 \times Range1.position.z : 1963 = 0 \quad (4.135)$$

4.5.7 Solution

La résolution du CSP nous permet d'obtenir une solution à notre requête initiale. La figure 4.5.7 présente la scène que nous avons obtenue pour cet exemple. Nous pouvons y voir représentés les trois groupes d'objets de façon à respecter les contraintes spécifiées. Nous avons dans le coin supérieur droit le réfrigérateur et la première table situés de part et d'autre du four. Du côté gauche de la figure, nous pouvons voir le lit à proximité de la table et de la chaise. De plus, nous pouvons confirmer que la machine à laver se situe à proximité du séchoir.

Afin de mieux illustrer les relations de type *Support*, la figure 4.5.7 présente chacun des groupes de plus près. Nous pouvons voir sur la première figure que les deux objets se retrouvent effectivement sur la première table. De plus, nous pouvons voir sur la seconde figure

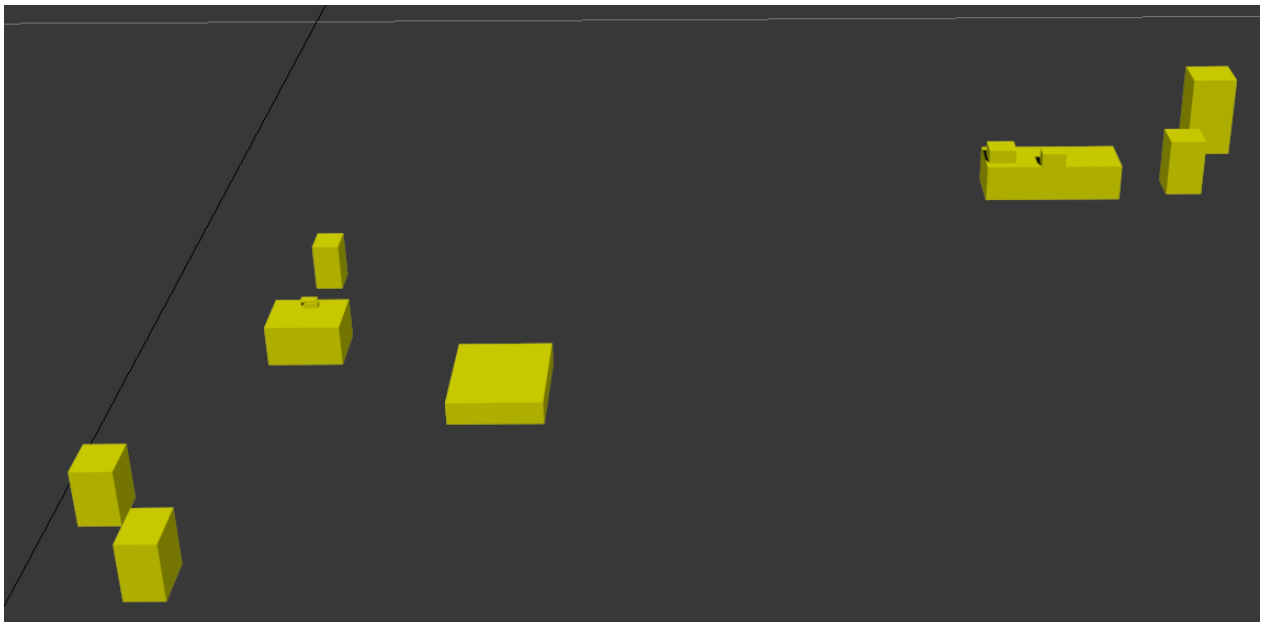

Figure 4.16 Disposition des objets dans la scène

que les trois livres ont été placés de manière à former une pile sur la table.

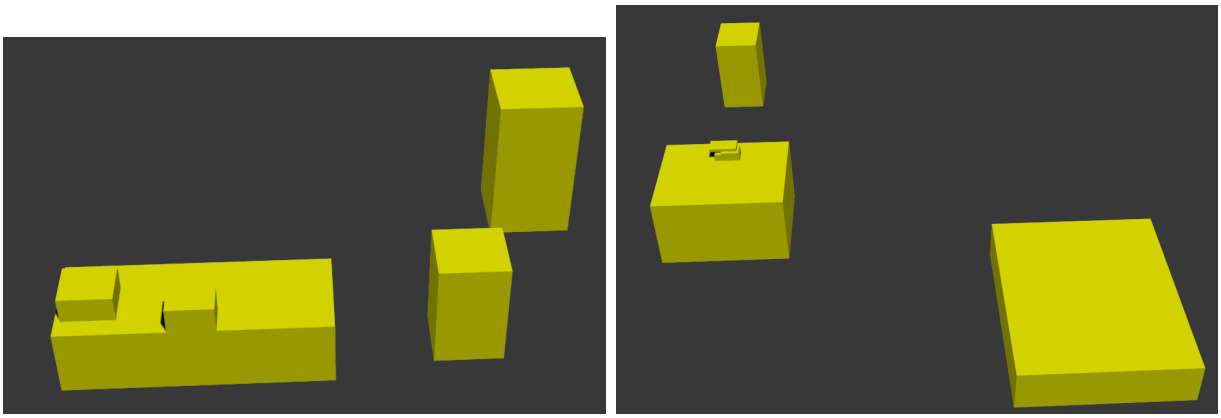

Figure 4.17 Disposition des objets reposant sur un support

CHAPITRE 5

MÉTHODOLOGIE

Dans ce chapitre, nous présentons la méthodologie de notre projet de recherche. Cette méthodologie présente les détails d'expérimentation nous ayant permis d'arriver à la solution proposée dans le chapitre 4.

Nous présentons tout d'abord l'architecture logicielle de notre système, accompagnée d'une description sommaire de chaque module, dans le but de comprendre le flot de données de notre application. Par la suite, nous expliquons certains détails de notre approche qui peuvent être approfondis davantage. Finalement, nous présentons la structure des résultats obtenus, ainsi que le plan d'expérience ayant mené à leur obtention.

5.1 Architecture logicielle

Notre architecture logicielle suit la structure d'un pipeline unidirectionnel permettant la génération d'une scène virtuelle. Le pipeline permet donc de passer de la représentation abstraite de scène à une scène 3D dans un format de représentation standard. Le schéma de notre architecture logicielle est présenté à la figure 5.1.

Notre pipeline implique plusieurs modules, formats et interfaces de programmation, et combine à la fois les langages Java et C++. Nous présentons rapidement les différents éléments de cette architecture.

Représentation abstraite

La représentation abstraite correspond à un fichier en format XML décrivant la scène à modéliser. Ce fichier correspond à la formulation de la requête décrite dans notre solution à la section 4.1.

Module de connaissance

Le module de connaissance gère les communications avec l'ontologie. Il permet d'interroger l'ontologie de manière à ajouter de l'information à la représentation abstraite. La communication avec l'ontologie se fait par le biais de requêtes SPARQL à l'aide de l'interface de programmation JENA. Le détail de ces requêtes est présenté dans la section 5.2.

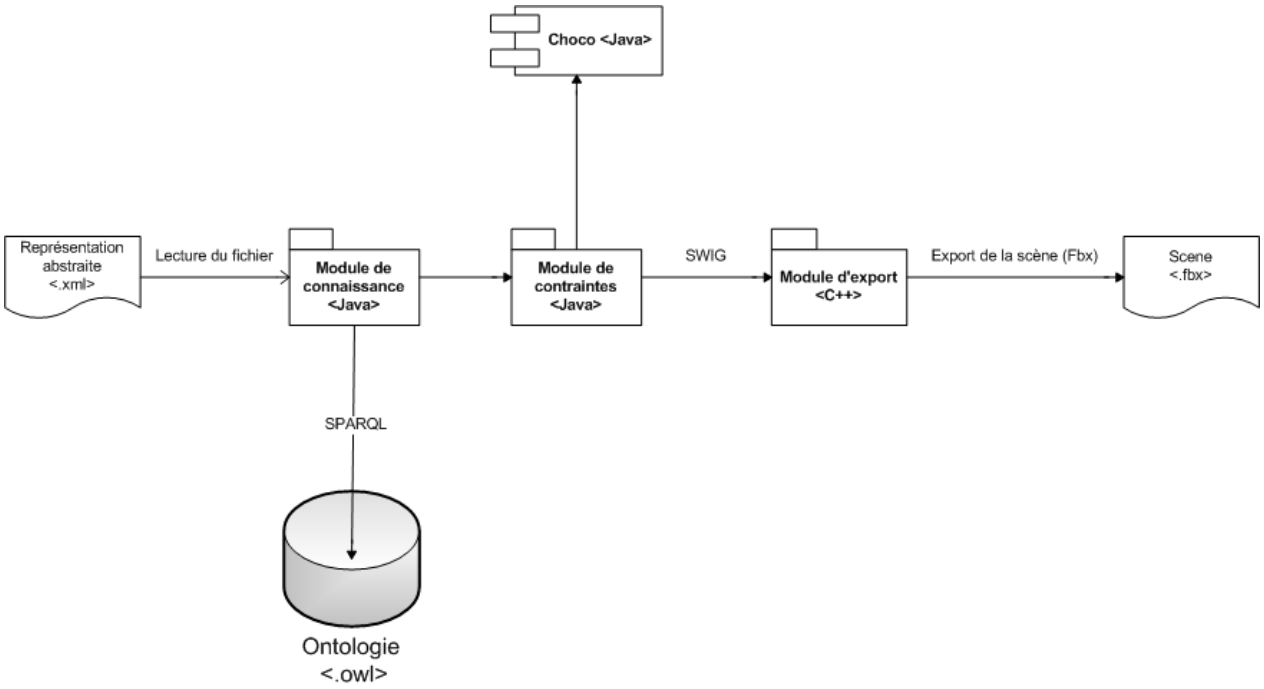

Figure 5.1 Architecture logicielle de l'outil expérimental

Module de contraintes

Le module de contraintes traduit la représentation abstraite en une modélisation de CSP. Il s'agit donc d'utiliser les informations issues de la base de connaissances pour définir des variables et des contraintes correspondant à la représentation abstraite initiale.

Par la suite, cette modélisation est fournie en entrée à un résolveur de contraintes qui sera utilisé pour obtenir une solution valide de la scène que nous voulons générer. Notre architecture repose sur l'utilisation du résolveur Choco (Jussien, N. and Rochart, G. and Lorca (2008)).

Communication entre Java et C++

L'utilisation combinée des langages de programmation Java et C++ implique l'utilisation d'une interface de programmation permettant de communiquer de l'information entre les deux langages. Dans notre architecture, cette communication repose sur l'utilisation de SWIG, qui nous permet de définir automatiquement le code Java permettant d'appeler une librairie C++.

Module d'exportation

Finalement, le module d'exportation est responsable d'exporter notre solution selon un format de représentation de scène multimédia. Ce module nous permet d'exporter notre scène dans le format Fbx, par le biais de l'interface de programmation Fbx d'Autodesk (Ansari (2011)).

5.2 Interrogation de la base de connaissances

La représentation de la connaissance par le biais d'une ontologie constitue un élément important de notre solution. En effet, c'est cette représentation qui nous permet de combler l'abstraction présente dans la représentation initiale et d'ainsi traduire notre représentation en un système de contraintes.

La section 4.2 nous a permis de présenter la structure de représentation interne de notre ontologie ainsi que l'information contenue à l'intérieur de celle-ci. Le but de cette nouvelle section est de présenter la méthodologie que nous avons utilisée pour interroger notre ontologie.

5.2.1 Structure d'une requête SPARQL

L'interrogation d'une ontologie repose sur la construction de requêtes *SPARQL*, un langage de requêtes comparable au langage *SQL*, permettant de rechercher et de manipuler des données représentées dans le format *RDF*.

5.2.2 Requêtes implémentées

Dans le but de compléter la représentation abstraite de scènes, nous avons besoin de formuler deux types de requêtes pour interroger notre ontologie. D'une part, nous voulons être capable d'obtenir les valeurs numériques associées à une classe d'objets, et d'autre part nous voulons être capable de connaître l'objet d'une relation impliquant un objet.

Recherche des attributs numériques d'une classe

La section 4.2.1 présentait notre hiérarchie d'objets. Afin d'être en mesure de représenter correctement ces objets dans l'espace, chaque classe d'objets possède des attributs numériques caractérisant leur masse et leurs dimensions.

Ainsi, le premier type de requête que nous considérons implique d'aller lire les propriétés numériques d'un objet dans l'ontologie. Cette requête nécessite de connaître l'objet et la

relation considérée. Le résultat obtenu est l'intervalle définissant la valeur possible de la propriété.

Objet d'une relation

La section 4.2.2 présentait les différentes relations ayant été modélisées dans notre ontologie. La plupart de ces relations permettent d'ajouter un nouvel objet à la représentation abstraite.

Le second type de requête que nous considérons implique d'identifier l'objet d'une relation par rapport à un objet. Cette requête nécessite de connaître l'objet et la relation considérée. Le résultat obtenu nous indique le type de l'objet qui est relié à notre objet par la relation donnée.

Par exemple, si nous effectuons ce type de requête en considérant le type *Book* et la relation *hasSupport*, nous obtiendrons comme résultat la classe d'objets *Furniture*.

5.3 Absence de rotations dans la méthode de résolution

Initialement, nous avions l'intention d'incorporer la rotation des objets à notre méthode de résolution. L'objectif était de pouvoir comparer une approche reposant sur une représentation abstraite avec les systèmes qui traitent les rotations dans leur méthode de résolution.

Cependant, nous avons rencontré plusieurs difficultés par rapport au traitement des rotations dans notre méthode de résolution. Cette section vise à présenter les détails de ce problème. Nous expliquons dans un premier temps ces difficultés. Par la suite, nous présentons les deux approches que nous avons tentées de modéliser pour surmonter ces difficultés.

5.3.1 Difficultés de la modélisation des rotations

La difficulté de représenter les rotations dans notre méthode de résolution est une conséquence directe du fait que la modélisation de nos variables repose sur des représentations entières. En effet, la rotation de vecteurs dans l'espace implique des transformations trigonométriques et nécessite l'utilisation des opérateurs $sin(x)$ et $cos(x)$.

Cependant, *Choco* ne fournit pas d'opérateurs trigonométriques supportant des valeurs entières. Cela est justifié puisque des valeurs entières à l'intérieur du domaine $[-1 : 1]$ ne fourniraient pas une précision acceptable. De plus, la modélisation d'un problème de contraintes en *Choco* ne nous permet pas de convertir des variables sur un domaine entier en des variables sur un domaine réel. Il n'est donc pas possible d'appliquer les opérateurs trigonométriques sur nos variables en les convertissant en variables réelles.

La modélisation d'objets tournés serait toutefois possible dans le domaine des entiers. En effet, si nous utilisons une précision suffisante, il est possible de représenter les sommets d"un rectangle ayant subi une rotation, par une approximation acceptable. La difficulté se situe donc au niveau du calcul de ces valeurs.

La solution à ce problème serait de définir de nouveaux opérateurs à l'intérieur de *Choco* et de les incorporer à la résolution du CSP. Cependant, le but de notre projet étant d'évaluer l'application d'une représentation abstraite à la planification spatiale, nous avons évalué que la définition de nouvelles méthodes de résolution à l'intérieur de *Choco* est hors du cadre du projet de recherche. Pour cette raison, nous avons exploré des solutions reposant sur les opérateurs offerts par *Choco*.

5.3.2 Solutions explorées pour la modélisation des rotations

Nous avons exploré deux solutions pour tenter de modéliser des rotations avec des variables entières. Dans un premier temps, nous avons tenté de combiner des variables entières et réelles dans notre modélisation. Par la suite, nous avons tenté de représenter les rotations par des rapports entre deux variable entières.

Combinaison de variables entières et réelles

Notre première approche a été d'implémenter une modélisation combinée de variables entières et de variables réelles. Ainsi, nous avions défini deux ensembles de variables présentant dans un cas des variables entières et dans l'autre cas des variables réelles. La représentation des objets reposait alors à la fois sur un vecteur réel et sur un vecteur entier.

Le principe était alors d'appliquer une contrainte d'égalité sur chaque paire de variables redondantes. Il nous était alors possible d'utiliser les variables réelles pour les calculs intermédiaires impliquant les rotations, tout en étant assurés d'obtenir une valeur entière à la fin de la résolution.

Cette approche semblait prometteuse au moment de l'implémentation, puisque nous avons été en mesure de modéliser l'ensemble du problème en combinant des variables entières et réelle. Par contre, au moment de la résolution, notre résolveur n'a jamais été capable de trouver une solution respectant notre modélisation, même après plusieurs heures de résolution, pour des problèmes impliquant très peu de variables.

Représentation des rotations par des rapports entiers

Suite à notre premier échec, nous avons tenté de modéliser une solution n'impliquant que des variables entières. Le principal obstacle était de réussir à représenter les valeurs

correspondant à l'application d'une transformation de rotation sur un vecteur.

L'approche que nous avons considérée était de représenter les valeurs réelles par un rapport de deux variables entières. Pour le calcul de ces rapports, nous avons pu implémenter des opérateurs $sin(x)$ et $cos(x)$ reposant sur des séries de Taylor.

Contrairement à l'approche par combinaison, cette approche nous a permis d'obtenir des solutions. Cependant, la précision de nos opérateurs n'était pas assez bonne pour supporter jusqu'à 180 degrés de rotation.

5.4 Configuration de Choco

La résolution du CSP correspondant à la scène devant être générée à partir de la représentation initiale a été réalisée par le résolveur de contraintes *Choco*. Cette section présente la façon dont nous avons configuré le résolveur dans le cadre de notre expérimentation.

5.4.1 Stratégie de branchement

Toutes les expériences ont été effectuées en utilisant la stratégie de branchement *DomOverWDegBinBranchingNew*, définie par Boussemart *et al.* (2004). Cette stratégie repose sur un branchement binaire. Lors d'un branchement, une variable est sélectionnée, et une valeur est attribuée à cette variable. Dans la première branche, la valeur sélectionnée est assignée à la variable. Dans la seconde branche, la valeur est supprimée du domaine de la variable.

Cette stratégie de branchement attribue dynamiquement un poids à chaque variable en fonction de la taille du domaine et du nombre de contraintes non résolues associées à chaque variable. Le poids associé à chaque variable permet de choisir la variable considérée par la stratégie de branchement de façon à accélérer la recherche.

5.4.2 Sélection des valeurs

La sélection des valeurs pour une variable lors d'un branchement est effectuée par une sélection de type *RandomIntValSelector*. Ce type de sélection choisit une valeur aléatoire à l'intérieur du domaine de la variable.

Nous utilisons une méthode de sélection aléatoire dans le but d'augmenter le niveau de réalisme des scènes générées. Ainsi, les objets sont placés selon une position aléatoire mais légale, et évite de générer une scène identique pour plusieurs exécutions à partir de la même représentation abstraite.

5.5 Structure des résultats

L'architecture logicielle présentée à la section 5.1 présente le pipeline de notre outil expérimental. Cet outil nous permet de générer une scène virtuelle à partir d'une description initiale selon notre format de représentation abstraite.

Chaque requête traitée avec succès par notre outil expérimental constitue donc une scène structurée selon le format de représentation *Fbx*. Chaque scène produite est composée d'un ensemble d'objets configurés selon des positions numériques définies à l'intérieur d'un environnement 3D.

Le but de notre projet de recherche n'est pas simplement d'évaluer si nous sommes capables de produire une scène 3D automatiquement à partir de notre représentation abstraite. Nous avons également formulé comme objectif d'être capable d'évaluer les scènes produites par planification spatiale.

Pour cette raison, nos résultats expérimentaux visent à produire un grand nombre de scènes virtuelles à partir de représentation initiales et à évaluer chaque scène produite à partir de métriques clairement définies. Ces métriques sont présentées dans la section 5.6.

5.6 Méthodes d'évaluation sélectionnées

L'évaluation des scènes produites par planification spatiale vise à répondre à deux questions. Dans un premier temps, nous désirons évaluer la complexité maximale des scènes que nous sommes capable de générer à partir d'une représentation abstraite. Ensuite, nous voulons évaluer la validité des scènes ayant été générées avec succès.

5.6.1 Complexité maximale

Tel qu'il a été présenté lors de la revue de littérature, la planification spatiale constitue un problème d'une grande complexité. Elle peut facilement introduire un grand nombre de variables et l'espace de solutions grandit exponentiellement avec le nombre de variables. Pour cette raison, nous sommes intéressé à connaître les limites de notre système par rapport au nombre d'objets et au nombre de relations considérées.

Cette évaluation repose sur une seule métrique d'évaluation. Il s'agit d'évaluer si une scène a pu être générée à partir d'une représentation abstraite. Afin de découvrir les limites de notre système, il suffit d'appliquer cette métrique à la génération de scènes impliquant un nombre d'objets et de relations de plus en plus grand.

Afin de recueillir le plus de détails possibles, nous avons toutefois exprimé cette métrique selon plusieurs valeurs de succès ou d'échec. Ainsi, l'évaluation de la complexité d'une scène peut prendre les valeurs de résultats suivants :

Succès Une scène a pu être générée à partir de la représentation abstraite.

Aucune solution Le résolveur a évalué qu'il n'existait aucune solution pour la modélisation de scène spécifiée.

Interruption Une erreur est survenue lors de la génération de la scène.

5.6.2 Validation des scènes générées

Connaître la complexité maximale des scènes que nous pouvons générer est une chose, mais nous voulons également être capables de mesurer la validité de la configuration de ces scènes par rapport à la description initiale.

Pour ce faire, nous définissons les métriques d'évaluation suivantes :

Score global Le score global consiste à déterminer si la scène produite est un succès ou non. Cette métrique permet de comptabiliser le nombre de scènes respectant parfaitement la représentation abstraite.

Proportion d'objets ajoutés Proportion d'objets de la représentation abstraite ayant été ajoutés à la scène générée.

Proportion d'objets respectés Proportion d'objets de la représentation abstraite ayant été correctement positionnés dans la scène générée.

Proportion de relations respectées Proportion de relations de la représentation abstraite étant correctement représentées dans la scène générée.

Nombre de défauts de la scène Cette métrique vise à évaluer le respect du sens commun des scènes générées. Ainsi, elle mesure le nombre d'incohérences pouvant être relevées dans les scènes générées.

5.7 Plan d'expérience

Cette section décrit le plan d'expérience que nous avons suivi pour l'obtention de nos résultats expérimentaux. Nous présentons dans un premier temps les éléments de la solution ayant été évalués dans notre expérimentation. Par la suite, nous présentons la méthode ayant été utilisée pour la production des scènes générées. Finalement, nous présentons la méthodologie employée pour la collecte des résultats et l'évaluation des métriques à partir des scènes générées.

5.7.1 Fonctionnalités évaluées

Afin de faciliter la génération des scènes et la collecte des résultats, nous avons décidé d'évaluer un sous-ensemble de la solution que nous avons présentée dans le chapitre 4.

Dans un premier temps, nous limitons les représentations initiales à des configurations de type *SpatialLocating*. Ces configurations permettent de représenter des éléments immobiles et caractérisent leurs positions dans l'espace. Cette configuration constitue donc la configuration la plus adaptée à la description de scènes pour la planification spatiale. Cependant, le choix de nous limiter à cette configuration implique que nous ne traitons aucune configuration impliquant des actions, et que nous n'évaluons pas la capacité d'ajouter des objets correspondant à une action.

Pour ce qui est de la base de connaissances, nous avons incorporé la recherche des attributs numériques d'une classe à notre plan d'expérience, mais nous n'utilisons pas les relations dans le but de compléter la représentation abstraite. Ainsi, bien que notre ontologie soit structurée de manière à ajouter des éléments à la représentation abstraite, notre évaluation ne se base que sur la capacité de générer une scène correspondant à la représentation abstraite.

Finalement, notre évaluation implique les objets que nous avons définis dans notre base de connaissances. Ces objets correspondent aux objets présentés à la section 4.3.2.

5.7.2 Génération de scènes

Nous avons indiqué dans la section 5.6 que l'obtention de résultats impliquait d'évaluer des métriques sur un grand nombre de scènes générées. La génération d'un grand nombre de scènes implique également la création d'un grand nombre de requêtes de scènes.

Afin de faciliter la création des requêtes, et par souci de limiter le biais expérimental par rapport aux scènes évaluées, nous avons choisi d'utiliser une méthode de génération aléatoire pour la création de nos requêtes. Cette méthode consiste à créer des configurations de type *SpatialLocating* impliquant un nombre d'objets donné et un nombre de relations données. Les objets sont obtenus à partir de l'ontologie alors que les relations correspondent aux contraintes traitées par la solution.

Cette méthode de génération nous permet du même coup d'évaluer la complexité maximale des scènes que nous sommes capable de générer. Nous commençons par générer un nombre fixe de scènes comportant 1 objet et aucune relation, et nous évaluons le succès moyen de génération de ces scènes. Si le succès est supérieur à un seuil défini, nous générons un nouvel ensemble de scènes en augmentant le nombre de relations de 1. Nous continuons à augmenter le nombre de relations tant que le taux de succès est supérieur au seuil fixé. Une fois cette limite atteinte, nous augmentons alors le nombre d'objets et recommençons avec un ensemble de scènes n'impliquant aucune relation.

Les résultats expérimentaux ont été obtenus à partir de représentations abstraites générées aléatoirement. Nous générons 20 scènes par niveau, et nous évaluons ces scènes avec un seuil de succès de 0.50. Nous fixons le nombre maximal d'objets à 15 objets.

CHAPITRE 6

RÉSULTATS ET DISCUSSION

Ce chapitre présente les résultats obtenus par l'application du plan d'expérience décrit à la section 5.7. Nous présentons d'abord les résultats mesurant la complexité maximale pouvant être traitée, puis les résultats mesurant la validité des scènes que nous avons pu générer.

Suite aux résultats, nous présentons la discussion associée à notre expérience. Nous commençons par une analyse des résultats dans laquelle nous évaluons l'effet du nombre d'objets et du nombre de relations sur le taux de succès et la validité des scènes générées. Nous continuons notre analyse en évaluant les limitations de notre méthodologie par rapport aux conclusions que nous pouvons retirer de l'analyse de nos résultats. Par la suite, nous enchaînons en comparant la complexité de nos scènes à celle des approches présentées dans la revue de littérature.

Nous poursuivons la discussion en effectuant un retour sur les objectifs que nous avions définis au terme de la problématique. Ainsi, nous évaluons l'impact de l'utilisation d'une représentation abstraite pour la planification spatiale, ainsi que les effets de l'application d'une ontologie dans le cadre de la génération de scènes.

Finalement, nous discutons de certains aspects de notre méthodologie qui auraient pu être approfondis davantage. Nous revenons d'abord sur notre modélisation du problème. Puis, nous nous intéressons à nouveau à l'orientation visant à générer une scène préalablement adaptée pour la génération automatique d'animations.

6.1 Complexité des scènes

Nous avons été en mesure de générer des scènes comportant jusqu'à 15 objets. Le tableau 6.1 présente les mesures de complexité en faisant varier le nombre d'objets et le nombre de relations.

Pour chaque nombre d'objets considéré, le dernier nombre de relations correspond à la complexité maximale atteinte. La complexité maximale est l'étape durant laquelle nous n'avons pas été en mesure de respecter le taux de succès minimal choisi pour l'expérimentation. Pour cette raison, les taux de succès présentés dans ces tableaux peuvent être inférieurs à 50% quand il s'agit de la complexité maximale.

Les pourcentages nous permettent d'indiquer la proportion de scènes ayant été générées avec succès et la proportion de scènes échouées parce qu'il n'y avait aucune solution à la

Tableau 6.1 Complexité pour quelques étapes de 1 à 15 objets

Nombre d'objets	Nombre de relations	Ratio de succès	Ratio d'échec (Aucune Solution)
2	2	90.00%	10.00%
2	4	25.00%	75.00%
3	3	65.00%	35.00%
3	5	35.00%	65.00%
4	4	70.00%	30.00%
4	5	45.00%	55.00%
5	5	75.00%	25.00%
5	7	45.00%	55.00%
6	6	80.00%	20.00%
6	9	45.00%	55.00%
7	7	60.00%	40.00%
7	9	40.00%	60.00%
8	8	65.00%	35.00%
8	9	40.00%	60.00%
9	9	70.00%	30.00%
9	11	45.00%	55.00%
10	6	85.00%	15.00%
10	10	70.00%	30.00%
10	13	40.00%	60.00%
11	8	80.00%	20.00%
11	11	55.00%	45.00%
11	13	45.00%	55.00%
12	8	80.00%	20.00%
12	12	55.00%	45.00%
12	14	35.00%	65.00%
13	10	70.00%	30.00%
13	13	70.00%	30.00%
13	15	20.00%	80.00%
14	10	70.00%	30.00%
14	14	50.00%	50.00%
14	16	40.00%	60.00%
15	10	75.00%	25.00%
15	15	60.00%	40.00%
15	16	40.00%	60.00%

requête formulée.

La figure 6.1 présente, pour chaque niveau, le taux de succès en fonction du nombre d'objets et du nombre de relations. La figure 6.2 présente le nombre maximal de contraintes atteint en fonction du nombre d'objets.

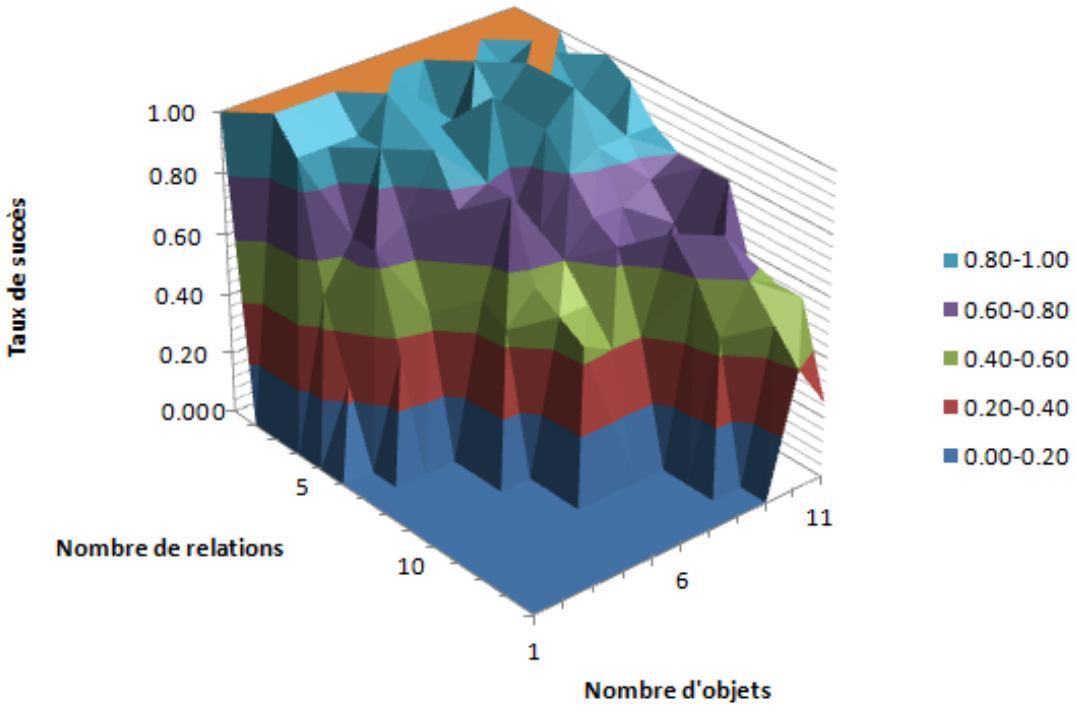

Figure 6.1 Taux de succès en fonction du nombre d'objets et du nombre de relations

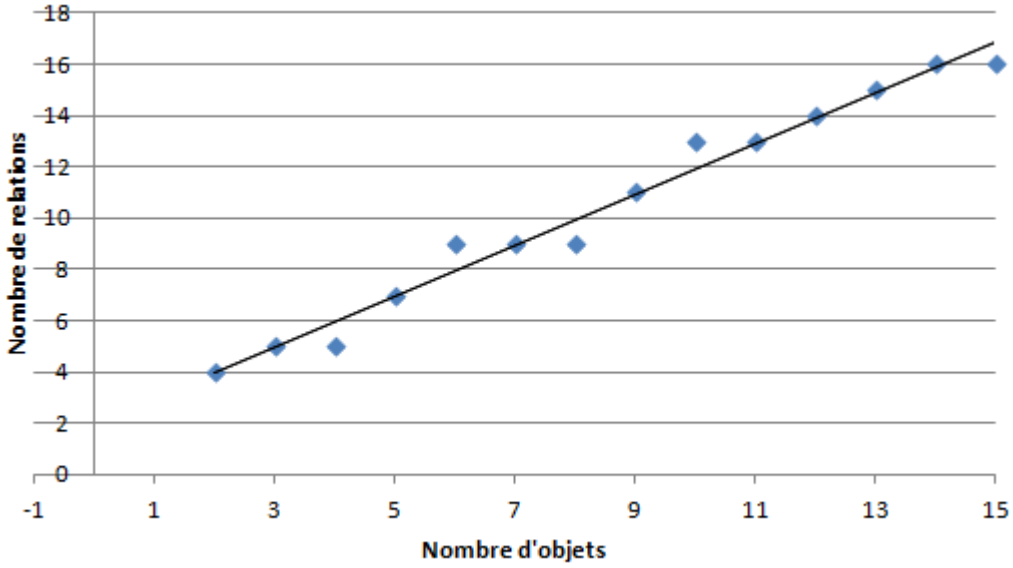

Figure 6.2 Nombre maximal de relations en fonction du nombre d'objets

6.2 Validité des scènes

Étant donné la rigidité de la modélisation de notre CSP, nous avons obtenu des taux de validité maximaux pour l'ensemble des scènes pour lesquelles nous avons été en mesure de générer une solution. Dans notre expérimentation, l'évaluation de la validité de la solution se situe donc au niveau de la résolution du CSP. Si nous ne sommes pas en mesure de générer une scène respectant parfaitement la représentation initiale, aucune solution n'est générée.

6.3 Analyse des résultats

Cette section présente notre analyse des résultats par rapport à la complexité et la validité des scènes que nous avons produites. De plus, nous présentons les observations que nous avons pu faire par rapport à la cohérence des scènes.

6.3.1 Analyse du taux de succès en fonction de la complexité des scènes

Le tableau 6.1 présente les résultats de notre expérience par rapport à la complexité des scènes générées. Ces résultats démontrent que nous avons été en mesure de générer des scènes contenant jusqu'à 15 objets et 16 contraintes. Cela nous indique que la planification spatiale à partir d'une représentation abstraite est réalisable pour des scènes de faible complexité.

Ces résultats nous permettent également d'observer l'effet du nombre d'objets et du nombre de relations sur la complexité de résolution des CSPs que nous modélisons à partir de notre représentation abstraite. D'après la figure 6.1, nous pouvons remarquer que le taux de succès se détériore lorsque l'on augmente le nombre de relations, mais nous pouvons traiter un plus grand nombre de relations lorsque le nombre d'objets est grand. Cela nous indique qu'il n'est pas nécessairement plus difficile de résoudre des scènes impliquant un grand nombre d'objets. En effet, la complexité se situe plutôt au niveau du nombre de relations considérées par rapport au nombre d'objets considérés. Ce comportement est logique puisque ce sont les contraintes qui apportent des limitations par rapport aux possibilités de placement des objets.

La figure 6.2 nous permet de mettre en relation le nombre de relations avec le nombre d'objets. Dans le cadre de notre expérience, nous pouvons remarquer que le nombre maximal de contraintes que nous avons pu traiter est généralement proche du nombre d'objets présents dans la scène. Étant donné le fait que nous considérons des contraintes binaires, cela nous indique qu'il devient très difficile de résoudre nos scènes lorsque chaque objet est soumis à plus d'une contrainte.

6.3.2 Analyse des cas sans solution

Le tableaux 6.1 présentait l'évolution des taux de succès par rapport à la capacité de notre méthode à générer des scènes de plus en plus complexes à partir de l'ensemble d'objets et de relations considérés. Cette section vise à présenter les cas observés par rapport à l'impossibilité d'obtenir une solution à partir de nos modélisations.

Relations de projection opposées

Les relations de projections sont par définition en opposition deux à deux. Ainsi, il est impossible pour un objet de respecter à la fois une relation de type *LeftProjectionExternal* et *RightProjectionExternal* ou une relation de type *BackProjectionExternal* et *FrontProjectionExternal*.

Relations de distance opposées

Similairement aux relations de projections, les deux relations de distance sont mutuellement exclusives. Ainsi, il est impossible pour une paire d'objets d'être contraints à la fois à une relation de type *Proximal* et une relation de type *Distal*.

Conflit entre la relation de support et les autres relations

Finalement, la relation de type *Support* ne permet pas d'être combinée à un autre type de relation de notre modèle. Dès qu'un objet impliqué dans une relation de support est soumis à une nouvelle relation, il est impossible de générer une solution à notre problème.

6.3.3 Observations par rapport à la cohérence des scènes produites

En plus des problèmes présentés dans la section précédente, la validation de nos scènes nous a permis de détecter quelques incohérences par rapport à la configuration des objets dans les scènes que nous générons. Ces incohérences ne constituent pas des erreurs par rapport à nos requêtes, mais plutôt des éléments qui ne sont pas très fidèles à la réalité. Ces incohérences auraient été l'objet de la métrique subjective, que nous avons choisi de ne pas traiter, en raison du biais de notre méthodologie d'évaluation. Nous avons tout de même noté les incohérences suivantes.

Positionnement des supports

Lorsque nous obtenons une solution pour une scène comprenant des relations de support, les relations de support correspondent à la formulation des contraintes. Ainsi, nous avons

toujours un objet placé sur son support de façon à ne pas en tomber. Cependant, le simple respect de ces contraintes ne suffit pas à produire des scènes réalistes. En effet, la façon dont nous plaçons des objets sur un support varie selon les objets considérés. De manière générale, nous évitons de placer des objets en périphérie d'un support, de façon à réduire les risques de chute. Par contre, il existe également des situations particulières, comme pour le placement de couverts sur une table, où nous visons à les placer en périphérie.

Afin de générer des scènes offrant un niveau de réalisme supérieur en ce qui concerne les supports, il serait intéressant d'améliorer la modélisation de cette relation à l'intérieur de la base de connaissance, de façon à mieux reproduire le sens commun associé au placement d'un objet sur un support. Cette relation pourrait avoir une position de placement préférée par rapport au centre du support, qui pourrait varier selon les objets considérés.

Espacement dans la scène

Dans toutes les scènes que nous générons, l'espacement des objets dans la scène constitue une incohérence marquée. En effet, même dans les cas où tous les objets respectent leurs contraintes de placement, ils peuvent être placés loin les uns des autres. Il serait plus réaliste de produire des configurations plus rapprochées. Une telle approche implique cependant qu'il serait plus difficile de respecter la contrainte d'espace entre les objets.

6.4 Limitations de la méthodologie

L'analyse de nos résultats nous a permis d'évaluer le succès de la génération automatique de scène par planification spatiale à partir d'une représentation abstraite. Cependant, nous avons identifié plusieurs limitations dans notre méthodologie qui nous empêchent de tirer certaines conclusions à partir de nos résultats. Cette section nous permet de présenter l'impact de ces limitations.

6.4.1 Rigidité de la modélisation

La modélisation des contraintes correspondant aux relations de notre représentation abstraite permet de placer les objets selon une position qui respecte totalement la représentation abstraite. Ces contraintes sont impliquées dans une méthode de résolution qui ne génère aucune solution lorsqu'il est impossible de respecter toutes les contraintes.

L'avantage de cette approche est que nous sommes certain de la validité des scènes générées lorsque nous obtenons une solution. Le désavantage est l'impossibilité d'obtenir une solution lorsque le problème est sur-contraint. Étant donné la nature du problème, il pourrait être

préférable d'obtenir une solution se rapprochant de la requête initiale lorsqu'il n'existe aucune solution.

Une telle approche nécessiterait une méthode de résolution capable de violer certaines contraintes lorsque le problème est sur-contraint. Ainsi, il serait possible d'obtenir une solution se rapprochant du résultat désiré. Il serait alors possible d'évaluer la validité des solutions produites à l'aide des métriques proposées à la section 5.6.2.

6.4.2 Variations de la génération aléatoire

Le fait de reposer sur une méthode de génération aléatoire nous a permis de traiter un grand nombre de cas, et d'éliminer une forme de biais au niveau de la création des cas expérimentaux. Par contre, cette méthode de génération introduit une forme d'incertitude par rapport aux scènes considérées.

La génération aléatoire introduit un certain nombre de contraintes binaires sur les objets de la scène. Ainsi, il est possible que dans certains cas les contraintes aient été distribuées assez uniformément entre les objets de la scène, alors que dans d'autres cas, il est possible qu'un même objet ait été soumis à plusieurs contraintes.

Cette méthode a pour conséquence que nous pouvons parfois nous retrouver avec des échecs dans des situations qui ne sont pas très complexes parce qu'un objet s'est retrouvé très contraint, et nous n'avons pas pu générer de solution. Nous pensons que cela peut expliquer le fait que le taux de succès ne varie pas de façon très uniforme en fonction de la complexité des scènes.

Il serait intéressant de répéter l'expérience plusieurs fois pour évaluer les variations introduites par la génération aléatoire. Il serait également intéressant de réévaluer le même jeu de données avec une solution capable de s'ajuster dans le cas d'un CSP inconsistant.

6.4.3 Absence de validation de l'ontologie

Une partie de notre solution constitue une base de connaissances sous la forme d'une ontologie. Le but de cette ontologie est d'être en mesure d'enrichir la requête pour générer une scène ayant un niveau de réalisme plus élevé en ce qui concerne le placement des objets. Cependant, nous n'avons pas évalué cet élément de notre solution dans notre expérimentation.

La raison principale de cette omission est la nature subjective de la validation de cette étape. En effet, il n'est pas facile de définir des métriques d'évaluation qui relèvent du sens commun. De plus, afin d'évaluer ces métriques de façon non biaisée, il serait préférable d'introduire un grand nombre de participants n'ayant pas de lien direct avec le projet de recherche.

Ainsi, la validation de l'ontologie se limite à des tests d'utilisation simple. Nous avons pu

valider que l'interrogation de l'ontologie par rapport à certains types de relation nous retourne les informations désirées. Cependant, la génération de scènes à partir de représentations abstraites enrichies par l'ontologie n'a pas été validée.

Nous pensons cependant que cette étape ne réduit pas la validation de la planification spatiale à partir d'une représentation abstraite. En effet, l'ontologie nous permettrait d'enrichir la représentation abstraite, mais ne génère pas un format de représentation différent. Ainsi, la représentation de la requête avant et après l'ajout d'informations par l'ontologie repose sur le même format.

6.5 Analyse de performance du générateur de scènes

Le but de cette section est de comparer les résultats que nous avons obtenus avec ce que nous connaissons des systèmes ayant été présentés dans la revue de littérature. Nous comparons ces résultats en tenant compte des objectifs spécifiques de notre projet de recherche, de notre modélisation du problème ainsi que de la méthode de résolution que nous avons utilisée.

6.5.1 Comparaison de performances

L'analyse des résultats nous indique que nous sommes capables de générer des scènes impliquant jusqu'à 15 objets et 16 relations. En comparaison avec les systèmes que nous avons présentés dans la revue de littérature, notre expérimentation ne vise pas le même ordre de grandeur en ce qui concerne la complexité des scènes condidérées. En effet, nous n'avons pas vérifié le comportement de notre système pour le placement de scènes impliquant jusqu'à 300 objets.

À partir de l'analyse de nos résultas expérimentaux, nous ne pensons pas que notre système aurait été en mesure de traiter des modélisations de scènes impliquant 300 objets à l'intérieur d'un temps raisonnable. Cela nous indique que la performance de résolution suite à notre modélisation n'est pas comparable à celle des systèmes sur lesquels nous avons basé notre expérience. Nous développons notre analyse par rapport à cette différence de performance dans le reste de cette section.

6.5.2 Orientations du projet

Nous pensons que la complexité que nous avons évaluée lors de notre expérimentation est suffisante pour répondre à nos objectifs expérimentaux. En effet, le but de notre projet de recherche est d'étudier la faisabilité d'utiliser une représentation abstraite pour la planification spatiale. Il est donc naturel que nos efforts aient été concentrés sur la création et l'application

de cette représentation et non sur la performance de génération. Pour cette raison, nous pensons que la performance que nous avons atteinte est suffisante pour valider l'utilisation de notre représentation abstraite dans le cadre de la planification spatiale. En effet, cette performance a été suffisante pour générer un grand nombre de scènes selon notre modélisation du problème et nous a permis d'amasser des données expérimentales pour l'analyse de notre méthode.

Par contre, il est évident que pour que notre approche soit réellement utilisable, il serait nécessaire de retravailler notre modélisation et notre résolution afin d'être en mesure de traiter des scènes impliquant un grand nombre d'objets. Pour cette raison, nous pensons qu'il serait intéressant de réaliser une nouvelle expérimentation suite à une optimisation de la méthode de génération de la scène. Une telle méthodologie nous permettrait alors de valider l'utilisation de notre représentation abstraite dans le cadre de problèmes de placement plus complexes. Nous présentons nos idées par rapport à cette optique de travail dans la section 7.1.

6.5.3 Améliorations de la performance

La source principale de la complexité de notre méthode de résolution se situe probablement au niveau de la modélisation de notre CSP. En effet, chaque objet est représenté par trois variables correspondant à sa position dans l'espace. Cependant, ces variables sont introduites dans le modèle de *Choco* en tant que variables complètement distinctes. Il pourrait être préférable que le lien entre ces trois variables soit pris en compte lors de la résolution.

Ainsi nous formulons une hypothèse selon laquelle l'atteinte d'une meilleure performance repose sur un meilleur couplage entre notre modélisation du problème et la méthode de résolution. Notre solution introduit beaucoup de complexité par rapport à la modélisation du problème, mais nous laissons la résolution à un outil externe qui n'est pas nécessairement orienté vers notre domaine d'application.

Il est connu dans la littérature que la planification spatiale implique une grande complexité par rapport à la modélisation du problème, en raison de la croissance exponentielle de l'espace de recherche. Pour cette raison, les systèmes que nous avons présentés dans le chapitre 3 définissent leur propre méthode de résolution pour la planification spatiale. L'amélioration de notre approche impliquerait donc l'utilisation d'une méthode de résolution adaptée à notre modélisation, dans laquelle nous pourrions définir des algorithmes de propagation de contraintes prenant en compte les variables et contraintes que nous avons définies.

6.6 Impact de l'utilisation d'une représentation abstraite

Suite à l'analyse de nos résultats dans les sections 6.3, 6.4 et 6.5, nous sommes en mesure de conclure qu'il est possible d'utiliser une forme de représentation abstraite en tant que format d'entrée pour un système de planification spatiale. En effet, nous avons représenté un grand nombre de scènes à l'aide de notre représentation et nous avons été capable de traduire ces scènes en un système de contraintes résoluble. À la lumière de ces résultats, nous discutons des particularités et des avantages qu'une telle représentation permet d'apporter à la planification spatiale.

6.6.1 Impact de l'utilisation de *GUM-Space*

L'utilisation de l'ontologie *GUM-Space* pour la représentation des relations spatiales s'est avérée un bon choix. En effet, l'ontologie offrait déjà une représentation pour la grande majorité des relations impliquées dans les représentation spatiales et la structure offerte permet de définir un format facile à comprendre pour la description initiale des scènes. De plus, cette représentation constitue un pas de plus dans la direction de la traduction automatique du texte en animations 3D, puisque *GUM-Space* a été définie avec l'objectif d'être utilisée pour le traitement de la langue.

Dans la perspective où notre représentation abstraite peut également être utilisée comme un format de script pour la description d'une scène virtuelle, nous admettons que le formalisme de *GUM-Space* est un peu moins adapté. Pour des scènes impliquant un grand nombre d'objets et de relations, l'expression des configurations dans un format XML devient rapidement difficile à lire et à comprendre. Pour une telle utilisation, il serait intéressant d'offrir une interface utilisateur plus intuitive qui aiderait à choisir des objets et des relations à intégrer à la représentation abstraite.

6.6.2 Lien entre la représentation abstraite et la base de connaissances

Notre représentation abstraite constitue également un excellent moyen d'intégrer une base de connaissances sous la forme d'une ontologie dans la structure de la planification spatiale. L'abstraction définie par chaque objet ou chaque relation peut facilement être reliée à un concept de notre ontologie. Ces liens permettent alors de combler l'abstraction et de transformer la représentation abstraite en un système de contraintes.

6.7 Application d'une ontologie dans le cadre de la génération de scènes

Notre méthodologie nous a également permis d'évaluer la possibilité d'utiliser une ontologie comme forme de représentation de la connaissance et d'appliquer cette ontologie à la planification spatiale à l'aide d'une représentation abstraite. Nous avons été en mesure de créer un modèle d'ontologie selon un domaine limité et d'utiliser des requêtes pour aller interroger cette ontologie. Nous faisons une revue des avantages que nous procure l'utilisation d'une ontologie et nous soulevons des éléments qui restent à explorer.

6.7.1 Avantages d'une ontologie pour la génération de scènes

Notre modélisation nous a permis d'obtenir plusieurs types d'informations permettant de compléter notre représentation abstraite et d'assister la planification spatiale. Nous avons été capable d'interroger l'ontologie dans le but d'ajouter des objets à la représentation abstraite, d'ajouter des contraintes de placement à certains objets, ainsi que de vérifier qu'une scène contient tous les éléments nécessaires à la réalisation d'une action.

Concrètement, l'ontologie nous permet de construire une représentation de sens commun adaptée à la planification spatiale. Elle peut renfermer l'ensemble des informations qui sont toujours vraies à propos des objets et de la façon de les placer dans l'espace. Notre modèle constitue une représentation plus adaptée à ce domaine que les représentations de sens commun habituelles, car il modélise des relations spécifiques à la planification spatiale, telle que la relation de support.

6.7.2 Types de représentation à explorer

Il serait également intéressant que notre ontologie puisse étendre le concept de sens commun pour représenter de l'information qui est souvent présente dans notre monde, sans toutefois être toujours vraie. Par exemple, on ne place pas les chaises de la même manière dans une salle d'attente que dans une salle à manger. Similairement, les objets présents dans un décor du temps des fêtes ne sont pas les mêmes que le reste de l'année.

Nous pensons qu'il serait possible d'organiser ce type de connaissance sous les concepts de *contexte* et d'*environnement*. Un *contexte* décrirait une situation particulière, comme le temps des fêtes, dans laquelle de nouvelles règles pourraient venir s'appliquer au placement des objets. Similairement, un *environnement* décrirait un lieu, comme une salle d'attente, dans lequel des règles particulières pourraient s'ajouter ou remplacer les règles habituelles.

Nous pensons que ces deux concepts constituent des intuitions de représentation de connaissance intéressantes qui pourraient être étudiées dans le cadre de la planification spatiale. Il serait donc intéressant d'étendre notre modélisation de la connaissance pour y intégrer

ces concepts, et d'évaluer leur efficacité par rapport à la complétion de requêtes à partir de notre représentation abstraite.

6.7.3 Options pour la création d'une ontologie d'envergure

En plus des nouvelles idées de connaissance à représenter dans notre modèle d'ontologie, nous avons également à répondre à la question de la création d'une ontologie d'envergure pour la planification spatiale. En effet, nous avons défini un petit modèle sur un domaine limité, ce qui nous a permis d'évaluer l'application d'une ontologie à la planification spatiale. Maintenant que nous avons jugé qu'il s'agit d'un format de représentation bien adapté à ce problème, le principal défi consiste à la création d'une ontologie impliquant une grande quantité de concepts tout en étant orientée pour la planification spatiale. Nous évaluons en détail les différentes options pour répondre à cette question dans la section 7.2.

6.8 Étendue de la modélisation du problème

Notre méthodologie visait à étudier la faisabilité d'utiliser une représentation abstraite comme format d'entrée pour la planification spatiale, en intégrant le concept d'ontologie comme format de représentation des connaissances. Bien que nous ayons été en mesure de valider la faisabilité d'une telle approche, nous l'avons fait en considérant un sous-ensemble des concepts connus de la modélisation d'un problème de planification spatiale. Ainsi, il ne nous est pas possible de juger de la faisabilité de l'utilisation d'une représentation abstraite de scène pour une modélisation complète de planification spatiale. Nous revenons sur les concepts que nous n'avons pas été en mesure d'évaluer.

6.8.1 Génération de scènes avec rotations des objets

Notre modélisation initiale des variables impliquait des objets décrits par une position et une orientation en trois dimensions. Au terme de notre expérimentation, nous n'avons été en mesure de produire que des scènes dans lesquelles les objets étaient orientés parallèlement aux axes directeurs des coordonnées de référence. Les limitations de l'intégration des rotations à notre expérimentation sont détaillées dans la section 5.3.

Tel que mentionné lors de l'explication du problème, la solution qui nous semblerait la plus efficace pour une planification spatiale avec support des rotations repose sur des contraintes spécialisées. La définition de telles contraintes permettrait d'appliquer une rotations sur des variables entières. Une telle solution implique également de définir l'impact que ces contraintes auraient sur le domaine des variables impliquées, et la façon de les résoudre au coeur de la résolution du CSP.

Sans avoir recours à des méthodes de résolution spécialisées, il est possible de considérer des solutions permettant de représenter des rotations partielles. La solution la plus simple à ce niveau est de permettre les rotations à angle droit. De cette façon, les objets considérés demeurent alignés avec les axes directeurs. Cette solution serait également supportée par *geost*, qui permet de définir plusieurs formes associées à un même objet physique.

Finalement, il pourrait être possible d'approximer une solution en traitant la rotation séparément de la translation dans la méthode de résolution. De cette façon, il serait possible d'utiliser une méthode de résolution différente pour le traitement des translations et le traitement des rotations. Ainsi, les objets pourraient d'abord être placés dans l'espace, puis tourner pour faire face à la bonne direction, ou bien être placés dans la scène en ayant déjà une direction prédéterminée. Il est évident qu'une telle approche comporterait un grand taux d'erreur pour le traitement d'objets longs et étroits, puisque la rotation a un grand impact sur l'espace physique utilisé par ces objets. Cette solution pourrait donc être une façon d'approximer une solution avec rotation, mais ne risque pas de donner de très bons résultats.

6.8.2 Éventail de contraintes considérées

Nous avons limité les contraintes impliquées dans notre modélisation à un petit ensemble. Le but de cette limitation était de ne pas trop alourdir la charge de travail associée à la création de notre méthode de résolution, alors que le plus important de notre recherche était relié à la représentation abstraite. Cependant, il serait intéressant d'évaluer notre représentation abstraite avec la totalité des contraintes que nous avons présentées dans la revue de littérature.

Nous pensons qu'il serait facile d'intégrer ces contraintes à notre représentation abstraite. En effet, la majorité de ces relations représentent des notions qui sont couvertes par le formalisme de *GUM-Space*. Il serait également facile d'intégrer ces concepts à notre représentation de connaissance. Il s'agirait simplement de définir de nouvelles relations et de créer les liens nécessaires à la représentation des objets impliqués dans de telles relations.

La plus grande difficulté de traiter une grande variété de contraintes se situe plutôt au niveau de la modélisation du problème. En effet, il est facile de définir de nouveaux types de relations dans un formalisme de représentation. Cependant, il peut être plus compliqué de définir la traduction de ces relations en contraintes mathématiques. De plus, l'intégration d'un grand nombre de contraintes différentes alourdit grandement la résolution du problème.

Dans le cas de notre expérimentation, nous avons dû limiter la complexité des contraintes que nous voulions intégrer à notre modélisation. En effet, lorsque nous avons tenté de résoudre des scènes impliquant des contraintes plus complexes que celles que nous avons présentées dans notre modélisation du problème, nous n'avons pas été en mesure d'obtenir des résultats

concluants. Dans certains cas, nous n'étions pas en mesure d'obtenir une solution après plusieurs heures de résolution. Dans d'autres cas, notre résolveur n'était tout simplement pas capable de traiter les modélisations et générait une erreur.

Le traitement d'une grande variété de contraintes serait donc intéressant pour permettre de valider complètement l'approche de la représentation abstraite pour la planification spatiale. Cependant, il serait d'abord nécessaire de réduire la complexité de notre modélisation du problème, ou d'opter pour une méthode de résolution mieux adaptée à la façon dont nous modélisons notre problème.

6.9 Génération d'une scène facilitant les animations

Le but du projet *GITAN* est d'être en mesure d'effectuer une traduction automatique du texte en animations 3D. Pour cette raison, l'une des orientations de notre génération de scènes était de faciliter la création d'animations 3D à l'intérieur de la scène générée. Bien que certaines mesures aient été prises en tenant compte de cette orientation, il reste du travail à faire pour supporter complètement la création des animations dans des scènes générées automatiquement.

6.9.1 Support complet des actions de *GUM-Space*

La définition de notre format de représentation abstraite repose sur le formalisme de *GUM-Space*. L'un des avantages de cette représentation provient du fait que les configurations permettant de spécifier des situations impliquent des actions. Ainsi, il est possible de décrire des scènes contenant des actions et d'utiliser la base de connaissances pour ajouter les objets nécessaire à ces actions dans la scène.

Cependant, nous avons décidé de limiter nos expérimentations à des configurations de type *SpatialLocating*, des scènes n'impliquant aucune action. Nous avons donc un format de représentation très bien adapté à la description de scènes orientées pour la génération d'animations. La prochaine étape est de supporter ces actions et d'être capable de générer automatiquement des animations à l'intérieur de nos scènes.

6.9.2 Traitement des contraintes de perception

Parmi les contraintes que nous n'avons pas considérés dans notre modélisation, certaines seraient particulièrement bien adaptés à la création d'animations dans une scène générée. Il s'agit des contraintes de perception décrites dans la revue de littérature. Ces contraintes permettent d'indiquer qu'une personne doit être en mesure de voir ou de circuler vers un objet et permettent de spécifier de l'espace nécessaire à l'utilisation de certains objets.

Dans l'optique ou la majorité des animations dans une scène impliquent les déplacement d'un être humain, ces contraintes constituent une information très intéressante à modéliser. L'intégration de ces contraintes à un système de génération de scènes permettrait donc de s'assurer qu'il est possible pour un agent d'atteindre les objets avec lesquels il veut interagir et qu'il dispose d'assez d'espace pour pouvoir les utiliser correctement.

6.10 Atteinte des objectifs

Au terme de notre expérimentation, nous pensons être en mesure d'affirmer que nous avons atteint nos objectifs de recherche. En effet, nous avons été en mesure de spécifier un format de représentation abstraite de scènes reposant sur l'ontologie *GUM-Space* et contenant les informations essentielles à la planification spatiale. Nous avons pu spécifier une représentation sémantique sous la forme d'une ontologie décrivant les concepts et relations nécessaires à la planification spatiale. Nous avons été capable d'utiliser ces représentations pour générer des scènes de complexité variées répondant aux descriptions initiales. Finalement, nous avons mesuré la complexité et la validité de nos scènes à partir de métriques que nous avons proposées.

L'atteinte de ces objectifs nous place en position de répondre à nos hypothèses et à notre question de recherche. Nous savons qu'il est possible de créer une représentation abstraite contenant les informations nécessaires à la planification spatiale. Nous savons qu'il est possible d'utiliser une représentation de connaissance pour compléter cette représentation abstraite. Nous savons qu'il est possible de transformer cette représentation abstraite en un système de contraintes qui peut être résolu. Nous pouvons donc conclure qu'il est possible de générer des scènes par planification spatiale à partir d'une représentation abstraite.

En plus de répondre à notre question de recherche, l'analyse de nos résultats permet de nous orienter par rapport aux prochaines étapes reliées à l'utilisation d'une représentation abstraite pour la planification spatiale. En effet, nous ne pouvons pas dire que nous avons couvert tous les cas dans lesquels notre représentation abstraite pourrait être appliquée. Il serait intéressant d'étudier à nouveau cette approche avec une modélisation plus performante, une modélisation plus complète et une représentation de la connaissance de grande envergure. Ces orientations sont présentées en détail dans le chapitre 7.

CHAPITRE 7

TRAVAUX FUTURS

Ce chapitre présente les perspective de travaux futurs visant à poursuivre l'étude de la planification spatiale reposant sur une représentation abstraite. Nous étudions l'intégration d'une méthode de résolution spécialisée, le développement d'une ontologie de grande envergure, la complétion de la modélisation du problème et la création automatique d'animations à partir de texte

7.1 Spécialisation de la méthode de résolution

Notre solution introduit une modélisation qui se complexifie rapidement lorsque le nombre d'objets et le nombre de relations augmentent. Dans le but d'atteindre un haut niveau de performance et d'être capable de générer des scènes très complexes, il serait intéressant de spécialiser notre méthode de résolution en adaptant la propagation des contraintes à la façon dont nous représentons nos variables. Nous soulevons trois stratégies pour effectuer cette spécialisation.

7.1.1 Intégration d'une méthode de résolution existante

La solution la plus simple à l'intégration d'une méthode de résolution spécialisée est d'utiliser une méthode existante. En effet, les systèmes présentés dans la revue de littérature reposent tous sur leur propre méthode de résolution de contraintes pour la planification spatiale. Intégrer ces solutions constituerait une approche simple pour l'obtention d'une méthode de résolution efficace.

7.1.2 Extension de Choco pour la planification spatiale

Notre solution définit des abstractions qui encapsulent les variables et les contraintes utilisées par *Choco*. Pour cette raison, notre modélisation présente des avantages par rapport à la création du CSP à partir de la représentation abstraite, mais pas par rapport à sa résolution. Cependant, l'un des aspects intéressants de *Choco* est qu'il est possible d'intégrer de nouvelles variables, de nouvelles contraintes, et de nouveaux algorithmes à sa méthode de résolution.

Il nous serait donc possible de définir les variables et les contraintes que nous présentons dans notre solution comme de nouveaux types à l'intérieur de *Choco*, et de définir des stra-

tégies de propagation de contraintes spécialisées s'appliquant sur ces nouveaux types. Cette solution serait probablement la plus facile à réaliser à partir de notre expérience actuelle, puisque tout le reste de la méthodologie pourrait demeurer inchangé.

7.1.3 Développement d'une nouvelle méthode de résolution

Il est toutefois possible que l'extension de *Choco* s'avère compliquée ou limitative. La troisième solution serait donc de développer notre propre méthode de résolution et de l'appliquer à notre problème. Cette approche constitue certainement une charge de travail beaucoup plus élevée, mais pourrait être celle qui offrirait les meilleurs résultats. C'est l'approche qui a été favorisée par les travaux précédents en ce qui concerne la planification spatiale.

7.2 Développement d'une ontologie d'envergure pour la planification spatiale

Le modèle d'ontologie que nous avons présenté dans notre solution demeure un modèle très limité en ce qui concerne la quantité d'informations qui est modélisée. En effet, nous présentons une organisation hiérarchique et une structure de relations qui sont adaptées à la planification spatiale, mais nous ne modélisons que quelques objets à l'intérieur de cette représentation. Une représentation de la connaissance d'envergure devrait plutôt viser à représenter l'ensemble des concepts reliés à la planification spatiale qui sont observables dans la réalité. Nous présentons encore une fois trois approches pour le développement de cette ontologie.

7.2.1 Continuer de développer un modèle indépendant

La première approche consiste à étendre notre modèle en y intégrant un grand nombre de concepts. La logique derrière cette approche est que, bien qu'il existe de nombreuses ontologies présentant déjà un grand nombre de concepts similaires, le défaut des ontologies est qu'elles présentent souvent un biais vers leur domaine d'application. Ainsi, en développant une ontologie adaptée à la planification spatiale, nous avons besoin de représenter des concepts qui ne sont pas représentés dans les ontologies appliquées à d'autres domaines. L'intégration de ces concepts pourrait être nuisible ou inadaptée à d'autres domaines. Pour cette raison, le développement d'un modèle orienté vers la planification spatiale est une approche viable.

7.2.2 Créer des liens avec des ontologies existantes

La seconde approche consiste encore une fois à étendre notre propre modèle, mais en supportant des liens vers d'autres ontologies. Ainsi, nous pourrions utiliser une hiérarchie

d'objets similaires à une autre ontologie, et indiquer par des liens qu'il s'agit des mêmes objets. Il nous serait alors possible de représenter des concepts présents dans d'autres ontologies, mais de les relier par des relations qui sont propres à notre domaine.

7.2.3 Étendre un modèle de sens commun existant

La troisième et dernière approche implique d'abandonner le modèle que nous avons étudié dans notre projet, et d'étendre un modèle de sens commun existant. Les modèles de sens commun sont les représentations de connaissance qui se rapprochent le plus de la planification spatiale. Il s'agirait donc d'étendre ces modèles dans le but d'y ajouter les relations que nous avons considérées pour la planification spatiale.

Nous pensons que *ConceptNet* (Liu et Singh (2004)) serait le meilleur candidat d'extension pour supporter une représentation de la planification spatiale. En effet, la hiérarchie d'objets représentée se rapproche de notre représentation, et il existe déjà quelques relations similaires aux relations que nous considérons. L'extension de *ConceptNet* impliquerait d'ajouter les relations intervenant dans la planification spatiale, les actions que nous considérons et tout autre concept qui aurait besoin d'être représenté.

7.3 Complétion de la modélisation du problème

L'évaluation complète de la faisabilité de la planification spatiale à l'aide d'une représentation abstraite implique d'étendre l'étude à une modélisation complète du problème. Cette orientation implique tout d'abord d'améliorer la performance de notre modélisation, afin d'être capable de générer des scènes comportant des relations complexes. Une modélisation complète impliquerait des objets pouvant être translatés et tournés, et impliquerait l'ensemble des contraintes présentées dans la revue de littérature.

7.4 Création automatique d'animations à partir du texte

Notre projet de recherche visait à résoudre la génération automatique de scènes statiques, qui constitue une étape du projet de recherche *GITAN*, dont l'objectif est la traduction automatique de texte en langage naturel vers des animations 3D. Au terme de notre expérience, l'objectif de *GITAN* n'est pas encore atteint. Nous présentons dans cette section les travaux futurs en lien avec notre étape du problème.

7.4.1 Développement d'un analyseur de texte vers *GUM-Space*

Notre représentation abstraite repose sur le formalisme de *GUM-Space* pour la représentation des relations spatiales entre les objets. *GUM-Space* est une ontologie développée dans l'optique du traitement du langage naturel. Par contre, au moment de la réalisation de notre expérience, il n'existait toujours pas d'analyseur capable d'annoter du texte selon le formalisme de *GUM-Space*. Le développement d'un tel analyseur constitue donc une étape cruciale pour relier l'analyse du texte à notre méthode de résolution.

7.4.2 Génération automatique d'animations

L'étape suivant la génération de scènes statiques est la création automatique d'animations 3D à l'intérieur de la scène qui a été générée. Cette étape du projet *GITAN* fait actuellement l'objet d'un autre projet de recherche dont l'objectif est de générer un plan d'animation à partir de la représentation abstraite et de la scène statique. Au terme de ce projet, il restera à générer l'aspect visuel des animations à partir du plan qui aura été généré.

7.4.3 Application de modèles après la planification spatiale

La génération de la scène statique vise à résoudre le problème de planification spatiale des objets. Ainsi, nous produisons une configuration valide d'objets qui correspond à ce qui a été spécifié dans la représentation abstraite initiale. Cependant, cette configuration correspond à une disposition de volumes englobants qui représentent les objets de la scène.

Afin d'être capable de générer des scènes réalistes, il est important d'être en mesure de remplacer nos volumes englobants par des modèles 3D représentants fidèlement les objets de la scène. Cette étape implique d'avoir accès à une grande variété de modèles 3D et d'être capable d'identifier les modèles représentant chacun des concepts présents dans la scène. Une autre approche pourrait être de générer automatiquement la géométrie de chaque modèle à partir de connaissance associée à chaque objet.

7.4.4 Extension de la génération de scènes à d'autres contextes

Notre projet de recherche effectue une génération de scènes statiques reposant sur la planification spatiale. Cette technique de génération de scènes permet de placer des objets à l'intérieur d'une scène selon des contraintes de placement. Une telle technique est très bien adaptée pour la génération de scènes intérieures, dans lesquelles nous devons configurer correctement du mobilier et des objets variés.

La génération de scènes extérieures ou de scènes urbaines repose sur des techniques différentes de génération de scènes qui relèvent plutôt de la programmation procédurale. Pour

que *GITAN* soit en mesure d'animer des scènes dans des environnements variés, il serait intéressant d'intégrer ce type de génération de scènes au projet.

CHAPITRE 8

CONCLUSION

Le but de notre projet était d'évaluer la faisabilité de la planification spatiale en reposant sur une représentation abstraite de scène. Au terme de notre expérimentation, nous avons été en mesure de proposer une solution reposant sur une représentation abstraite, et nous avons été capable d'analyser et de comparer cette solution avec l'état de l'art en ce qui concerne la planification spatiale.

Notre solution repose sur la définition d'un format de représentation abstraite de scène s'appuyant sur le formalisme de l'ontologie *GUM-Space*. Cette représentation permet de spécifier une ou plusieurs configurations par scène, où une configuration décrit une situation dans laquelle se déroule une action. Une configuration nous permet d'introduire des objets dans la scène, et de définir des modalités de placement pour ces objets.

La représentation abstraite vise à être utilisée en tandem avec notre représentation sémantique de scène. Cette représentation constitue une ontologie dans laquelle nous avons défini une hiérarchie d'objets, de contrôleurs et d'actions. Cette hiérarchie définit des concepts qui peuvent être caractérisés par des relations. Cette ontologie nous permet de combler l'abstraction introduite par notre représentation abstraite en offrant la possibilité de formuler des requêtes pour permettre l'ajout d'objets à la scène ou l'ajout de contraintes aux objets à placer.

Notre expérimentation nous a permis de générer un grand ensemble de descriptions initiales de scènes, que nous avons transformées en des systèmes de contraintes qui ont pu être résolus pour former des configurations d'objets dans des scènes valides. Cette expérimentation nous a permis d'obtenir des résultats par rapport à la complexité maximale de scène que nous pouvions traiter, ainsi qu'à la validité des scènes que nous produisions. Nous avons été capable de générer des scènes impliquant jusqu'à 15 objets et 16 relations.

Au terme de l'analyse de nos résultats, nous avons été en mesure de valider nos objectifs et nos hypothèses de recherches. Nous avons donc pu conclure qu'il est possible de générer automatiquement une scène virtuelle par planification spatiale à partir d'une représentation abstraite. Cependant, nous n'avons pu valider cette hypothèse qu'à l'intérieur de certaines limitations. Une validation complète de la faisabilité de cette approche impliquerait de combler les limitations de notre expérimentation.

Notre modélisation introduit facilement des conflits entre les relations auxquelles peuvent être soumis les objets de la scène. Une telle modélisation en combinaison avec notre méthode

de résolution rigide fait qu'il est impossible d'obtenir une solution lorsqu'au moins un objet ne peut pas respecter les relations auxquelles il est soumis. Il serait probablement plus intéressant d'utiliser une méthode de résolution permettant la génération de solutions imparfaites dans le cas de problèmes sur-contraints, et d'évaluer la validité de ces solutions.

Notre expérimentation introduit également des limitations par rapport aux fonctionnalités de notre solution. Nous n'avons considéré qu'un sous-ensemble des configurations possibles de notre représentation abstraite, en nous limitant aux configurations n'impliquant pas d'actions. Nous n'avons pas validé la complétion de la représentation abstraite par la base de connaissances. De plus, notre modélisation du problème ne couvre pas la rotation des objets, ainsi que plusieurs types de relations qui sont habituellement considérées dans le cadre de la planification spatiale. Afin d'être en mesure de complètement valider l'utilisation d'une représentation abstraite, il serait intéressant d'étendre l'expérimentation pour couvrir toutes ces limitations.

Les travaux futurs relatifs à notre projet visent donc à combler les limitations de notre projet de recherche. Dans un premier temps, il serait intéressant d'améliorer la performance de la résolution à partir de notre modélisation du problème. Dans un second temps, il serait intéressant d'étendre notre modèle de représentation sémantique pour couvrir l'ensemble des concepts de la réalité applicables à la planification spatiale. Finalement, il serait intéressant d'étendre la modélisation du problème pour considérer les rotations des objets ainsi que tous les types de contraintes applicables à la planification spatiale.

Dans le cadre du projet GITAN, les travaux futurs visent à relier notre projet de recherche au reste du pipeline de traduction de texte en animation. Il s'agit donc de développer un analyseur de texte capable d'annoter du langage naturel selon le formalisme de *GUM-Space*, de générer automatiquement des animations à l'intérieur des scènes générées, et de représenter les objets à l'intérieur des scènes par des modèles 3D. Il s'agit également d'étendre la génération de la scène statique à des environnements extérieurs et urbains, qui reposent sur des techniques de génération différentes de la planification spatiale.

RÉFÉRENCES

ABACI, T., CIGER, J. et THALMANN, D. (2005). Planning with smart objects. *In The 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision : WSCG*. 25–28.

AKAZAWA, Y., OKADA, Y. et NIIJIMA, K. (2008). Automatic 3D scene generation based on contact constraints. *Proc. of the Eighth International*.

ANSARI, M. (2011). FBX Games Development Tools. *Game Development Tools*. 309–319.

APT, K. (2003). *Principles of constraint programming*. Cambridge Univ Pr.

BAKER, C. F., FILLMORE, C. J. et LOWE, J. B. (1998). The Berkeley FrameNet Project. *Proceedings of the 36th annual meeting on Association for Computational Linguistics* -, 86.

BATEMAN, J., HOIS, J., ROSS, R. et TENBRINK, T. (2010). A linguistic ontology of space for natural language processing. *Artificial Intelligence*.

BAYKAN, C. et FOX, M. (1991). Constraint satisfaction techniques for spatial planning. *Intelligent CAD Systems III, Practical Experience and Evaluation*. 187–204.

BELDICEANU, N., CARLSSON, M., PODER, E., SADEK, R. et TRUCHET, C. (2007). A generic geometrical constraint kernel in space and time for handling polymorphic k-dimensional objects. *Principles and Practice of Constraint Programming*, 180–194.

BESSIERE, C. (2006). Constraint Propagation. *Handbook of constraint programming*. 50–105.

BONNEFOI, P. et PLEMENOS, D. (1999). Object oriented constraint satisfaction for hierarchical declarative scene modeling. *WSCG*. Citeseer, vol. 99, 2–12.

BORNING, A. (1981). The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory. *ACM Transactions on Programming Languages and Systems*, 3, 353–387.

BOUSSEMART, F., HEMERY, F. et LECOUTRE, C. (2004). Boosting systematic search by weighting constraints. *ECAI*.

CHARMAN, P. (1993). Solving space planning problems using constraint technology. *ASI Constraint Programming : Students' Presentations*.

COYNE, B. et SPROAT, R. (2001). Wordseye : An automatic text-to-scene conversion system. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM New York, NY, USA, vol. ternetchap, 487–496.

ERICSON, C. (2005). *Real-time collision detection*. Morgan Kaufmann.

GLASS, K. et BANGAY, S. (2007). Constraint-based conversion of fiction text to a time-based graphical representation. *Proceedings of the 2007 annual research*, 19–28.

GUTIERREZ, M., VEXO, F. et THALMANN, D. (2005). Semantics-based representation of virtual environments. *International Journal of Computer Applications in Technology*, 23, 229–238.

HOMAYOUNI, H. (2000). A Survey of Computational Approaches to Space Layout Planning (1965-2000). *Citeseer*, 1–18.

IRAWATI, S., CALDERÓN, D. et KO, H. (2006). Spatial ontology for semantic integration in 3D multimodal interaction framework. *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications - VRCIA '06*, 1, 129.

JUSSIEN, N. AND ROCHART, G. AND LORCA, X. (2008). Choco : an Open Source Java Constraint Programming Library. *Proceedings of the 3rd International CSP Solver Competition held in conjunction with the 13th International Conference on Principles and Practice of Constraint Programming*. 7—-14.

KALLMANN, M. et THALMANN, D. (1998). Modeling objects for interaction tasks. *Proc. Eurographics Workshop on Animation and Simulation*.

KESSING, J., TUTENEL, T. et BIDARRA, R. (2010). Services in Game Worlds : A Semantic Approach to Improve Object Interaction. *Proceedings of the 8th International Conference on Entertainment Computing*. vol. 5709, 276–281.

LE ROUX, S. et GAILDRAT, H. (2003). Constraint-Based 3D-Object Layout using a Genetic Algorithm. *141.115.28.2*.

LENAT, D. B. (1995). CYC : a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38, 33–38.

LIU, H. et SINGH, P. (2004). Commonsense reasoning in and over natural language. *Knowledge-Based Intelligent Information and Engineering Systems*. Springer, 293–306.

MA, M. (2006). *Automatic conversion of natural language to 3D animation*. Thesis, University of Ulster, Faculty of Engineering.

MESEGUER, P., ROSSI, F. et SCHIEX, T. (2006). Soft Constraints. *Handbook of constraint programming2*. 302–349.

MILLER, G. A. (1995). WordNet : a lexical database for English. *Communications of the ACM*, 38, 39–41.

ORKIN, J. (2007). The restaurant game : Learning social behavior and language from thousands of players online. *Journal of Game Development*, 3, 39–60.

OTTO, K. (2005). The semantics of multi-user virtual environments. *Proc. of the Workshop towards Semantic Virtual Environments*. Citeseer, 1–10.

PETERS, C., MCNAMEE, B., DOBBYN, S. et O'SULLIVAN, C. A. (2003). Smart objects for attentive agents.

PFEFFERKORN, C. E. (1975). Problem Solving Design System for Equipment or Furniture Layouts. *Communications of the ACM*, 18.

SEEHOF, J., EVANS, W., FRIEDERICHS, J. et JJ (1966). Automated facilities layout programs. *Proceedings of the 1966 21st national conference*. 191—-199.

SHINYA, M. et FORGUE, M.-C. (1995). Laying out objects with geometric and physical constraints. *THE VISUAL COMPUTER*, 11, 188–201.

SUTHERLAND, I. E. (2003). Sketchpad : A man-machine graphical communication system. Rapport technique 574.

TUTENEL, T., BIDARRA, R., SMELIK, R. et DE KRAKER, K. (2009a). Rule-based layout solving and its application to procedural interior generation. *3D Advanced Media In Gaming And Simulation (3AMIGAS)*. 15.

TUTENEL, T., BIDARRA, R., SMELIK, R. M. et KRAKER, K. J. D. (2008). The role of semantics in games and simulations. *Computers in Entertainment CIE*, 6, 57.

TUTENEL, T., SMELIK, R., BIDARRA, R. et DE, K. J. (2009b). Using semantics to improve the design of game worlds. *Proceedings of the Fifth Artificial Intelligence for Interactive Digital Entertainment Conference*.

VAN HOEVE, W.-J. et KATRIEL, I. (2006). Global Constraint. *Handbook of constraint programming*. 190–229.

XU, K. (2001). *Automatic object layout using 2D constraints and semantics*. Thèse de doctorat.

XU, K., STEWART, J. et FIUME, E. (2002). Constraint-based automatic placement for scene composition. *Graphics Interface*. Citeseer, 25–34.